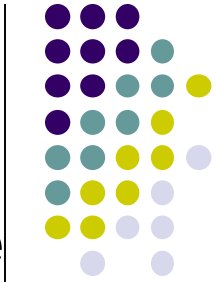
Développement d'applications mobiles

M2 RT/GLAR

Pr. El hadji M. NGUER

A propos de moi

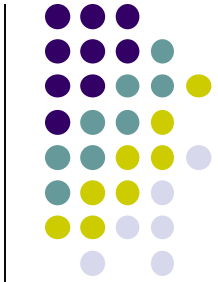


- Enseignant-Chercheur à **l'UFR SAT- Université de Saint-Louis**
- **Enseignements:**
  1. Algorithmique et programmation (Java, C, Pascal)
  2. Base de données (MySQL, PostgreSQL, XML)
  3. Développement web (XML; HTML5, CSS3, JavaScript, jQuery, Bootstrap, PHP, MySQL)
  4. Développement d'application mobile Android
  5. Théorie des langages et automates
- **Contact:**
  - **emnguer@ugb.edu.sn**
  - **UFR SAT, Université Gaston Berger BP 234 Saint-Louis.**

Sagesse chinoise...

«J'entends et j'oublie. Je vois et je me souviens. Je fais et je comprends.» ***Confucius***

Sommaire



1. Introduction générale
2. Préparer l’environnement de développement
3. Créez et exécutez votre première application Android
4. Concepts relatifs aux applications Android
5. Création d'interfaces graphiques
6. Intents et Services
7. Gestion des données
8. Multimédia
9. Distribuer et monétiser avec Google Play

Développement d'applications mobiles

INTRODUCTION GENERALE

M2 RT/GLAR

PARTIE 1

Pr. El hadji M. NGUER

L’omniprésence des technologies mobiles

Chiffres-clés

3,8 milliards d'utilisateurs mobiles dans le monde en 2008 qui ont générés 742 milliards de dollars de revenus. 6 milliards d'utilisateurs mobiles dans le monde en 2012.

366 milliards de chiffre d'affaires en 2011, selon l'ARTP au Sénégal pour 10,7 millions d'abonnés (88%), au 30 juin 2012.

Plus de 6 milliards d'abonné au mobile. "*A elle seule, la Chine compte 1 milliard d'abonnés, et l'Inde devrait elle aussi atteindre la barre du milliard en 2012*", relèvait l'UIT.

Plus d'un millier de nouveaux utilisateurs chaque seconde….

Il a fallu 20 ans pour franchir le cap du milliard d'utilisateurs mobiles et…

- **40 mois pour dépasser les 2 milliards**
- **24 mois pour atteindre les 3 milliards (juillet07)**

Des mobinautes exigeants

« Always On »…

- L'ordinateur personnel n'est plus l'outil par défaut de connexion à l'Internet,
- Le consommateur attend des offreurs de services que leurs contenus soient disponibles n'importe où, n'importe quand (« Everyone is in the Shop »),
- Ce sentiment s'accompagne d'un besoin d'instantanéïté (éviter les frustrations),
- Csq: Ces réalités conduisent à des comportements de surf mobile totalement différents de ce qu'on connaissait auparavant!



Quelques chiffres…

Le mobile est la première interface avec laquelle on interagit avant de s'endormir….

… et la première avec laquelle on se réveille!

- 71% des britanniques déclarent utiliser leur téléphone mobile comme réveille-matin,
- 25% des couples britanniques font parfois chambre à part à cause de l'addiction au téléphone mobile… ;-)))

Vers un environnement mobile « data-driven »

« The mobile device becomes people's primary interface to the Internet and social Networks »

- *Rudy de Waele Mobile Monday Amsterdam – sept 2008*

« Kakasenai Keitai« (*)

(*) indispensable mobile

Evolution des périphériques mobiles

- 1980-1990

Les technologies mobiles ou de téléphonie mobiles permettent une communication sans fil (téléphonique, Internet, ..) qui fonctionne même lorsque l'utilisateur est en mouvement.

A) Mobile Phones 1985-86: Motorola 8000S, Mitsubishi Roamer, Technophone PC135	A) Mobile Phones1987-1988: NEC 9A, Motorola 8500X, Nokia Cityman 1320, Philips PRC30E, British Telecom Coral (gauche à

Même si ça parait incroyable aujourd'hui, ces appareils étaient révolutionnaires à l'époque car ils permettaient de téléphoner sans être obligé d'être à côté du combiné.

Evolution des périphériques mobiles

- 1990-2000

Les premiers appareils de téléphonie mobiles de la décennie 1990 adressaient la question de la taille des appareils et de l'autonomie suffisante pour pouvoir émettre des appels durant toute la journée sans le recharger. Coté fonctionnalité c'était très basique avec seulement la possibilité d'enregistrer quelques numéros de téléphones. Mais le plus important est que cette décennie a vu le développement fulgurant de GSM qui signe la fin de l'ère analogique.

GSM phones 1993-4: Ericssn GH337, Motorola 5200, One2One M200, Nokia 2140

GSM phones 1995-6: Nokia 1610, Nokia 2110i, Nokia 8110, Motorola 7500, Ericsson GA318, Motorola StarTAC

GSM phones 1999: Motorola Timeport L7089, Ericsson T28, Nokia 8210, Motorola v3688, Nokia 3210, Segem 815

Evolution des périphériques mobiles

- **Depuis 2000**

ANDROID WEAR


PHONES


TABLETS


ANDROID TV


ANDROID AUTO

  - Des années 2000 à nos jours nous sommes dans la période de la convergence technologique et des objets connectés (diversification des équipements et des services).
  - Cette révolution est arrivée avec l'iPhone d'Apple, puis celle du système d'exploitation Android, enfin la création d'un ensemble d'équipement connectés comme les tablettes, les montres connectés, la télévision, …

Et le Sénégal dans tout cela ?

Au Sénégal ?


MOBILE SENEGAL
5 years of capacity and community building with impact
2008-2012
mobilesenegal.org
facebook.com/mobilesenegal
@mobilesenegal
mobilesenegal@gmail.com
© 2013 MobileSenegal

Au Sénégal ?

Optimisation de filière agricole, fluidification entre petits producteurs, fournisseurs, distributeurs, transformateurs, exportateurs financiers et clients.

Gamme de services métiers accompagnant le développement de l'accès à l'eau et à sa gestion responsable par les consommateurs et les opérateurs

Ecosystème de services spécialisés pour améliorer l'accès des populations aux infrastructures et aux services d'assainissement

plate-forme dédiée aux collectivités locales pour l'amélioration de leur gestion et le renforcement de leurs capacité à délivrer leurs services

Pour la maitrise des prix des denrées de première nécessité et de sauvegarde du pouvoir d'achat des ménages

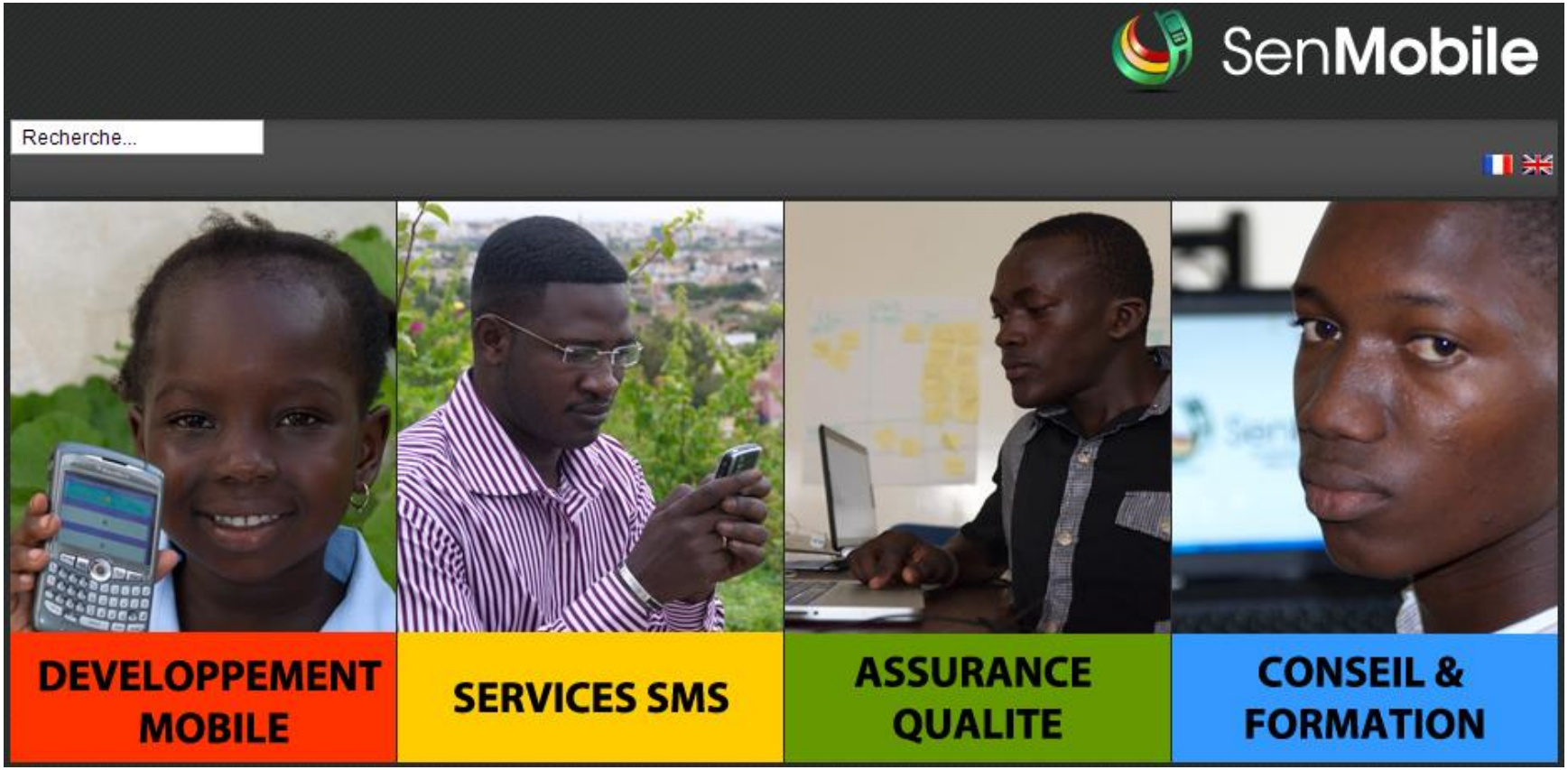
Services pour l'intégration économique et sociale des populations ou des segments de population les plus vulnérables

Outil d'optimisation pour des flottes de véhicules, pour améliorer les indicateurs clés du business routier en lien avec les SI de l'entreprise

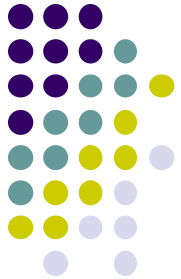
Solutions de convergence mobiles et géographiques destinées aux études, enquêtes et suivis de terrain dans un contexte de coût et de complexité maîtrisée

Ils nous ont fait confiance : La Banque Mondiale, le Gouvernement du Sénégal, Orange, l'UNICEF, l'USAID, Patisen, Ferrero...

Au Sénégal ?

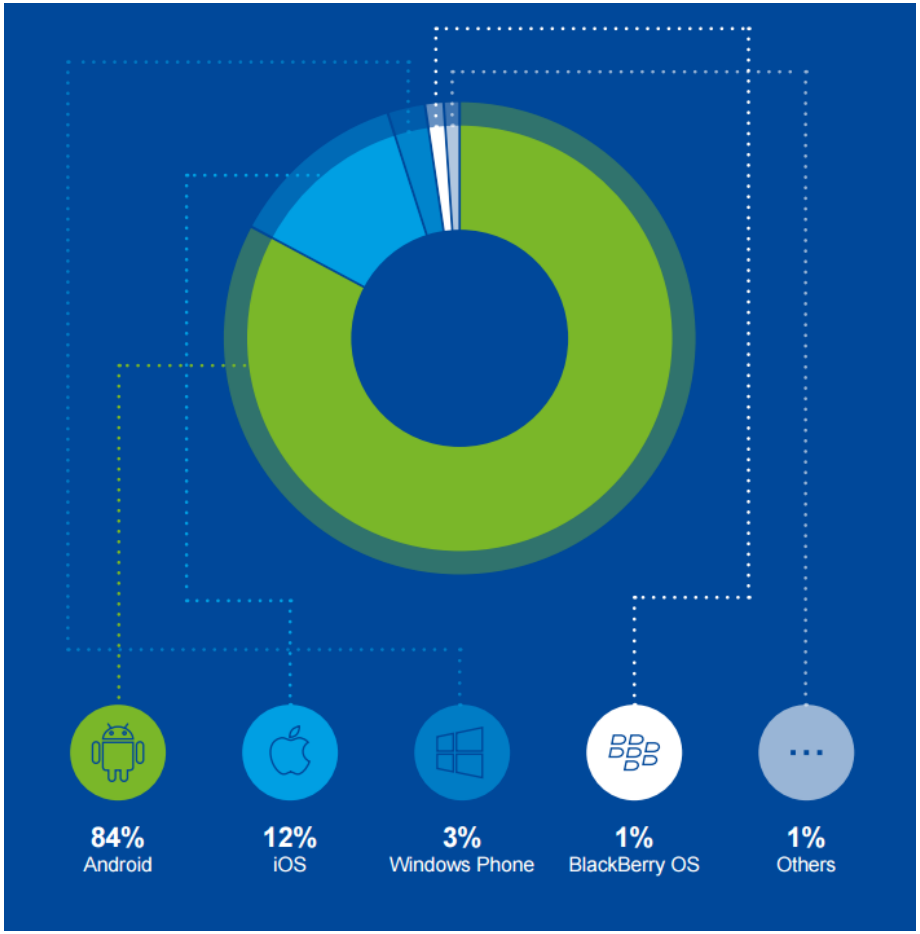

SenMobile
Recherche...
DEVELOPPEMENT MOBILE
SERVICES SMS
ASSURANCE QUALITE
CONSEIL & FORMATION

Système d'exploitation mobile



- **Un système d'exploitation mobile**, appelée aussi un système mobile, est un logiciel qui gère un appareil mobile intelligent tel qu'une tablette ou un smartphone.
- Les systèmes d'exploitation mobiles modernes combinent les caractéristiques d'un système d'exploitation d'un ordinateur personnel avec d'autres fonctionnalités comme un écran cellulaire, …

Mobiles OS Market (source http://www.idc.com/prodserv/smartphone-os-market-share.jsp)



Stratégies mobiles: Quelles plateformes?

Pour développer des applications mobiles, il est nécessaire de comprendre les principes sous-jacents de plates-formes. Ces plates-formes mobiles offre l'OS de base un environnement de développement et un market pour publier ces applications. Les principales plateformes mobiles sont : **Android, iOS, Windows Phone, BlackBerry OS, …**

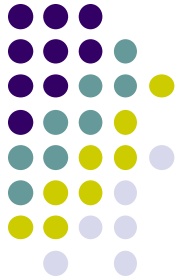
symbian

palm™

Windows Mobile

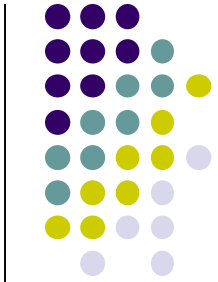
RIM®

Choix de la plate-forme mobile



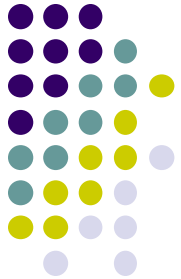
- La première étape dans le développement d'applications mobiles est de décider quelle plate-forme pour développer.
- Les paramètres suivants vous permettent de choisir celle qui est la plus adaptée:
  - **Le public cible:** En fonction du niveau de diffusion de votre application, vous pouvez choisir la plateforme dominante sur le marché;
  - **Marché:** Vous pouvez choisir votre application en comparant les parts de marchés de l'application.
  - **Caractéristiques spécifique:** Chaque plateforme a des caractéristiques spécifiques qui peuvent être utilisés pour votre application. Considérer les caractéristiques des dispositifs mobiles lors de la conception de votre application peut permettre de gagner du temps de développement, d'avoir une meilleur sécurité, ...

Principales plateformes mobiles en 2017



Mobile Plateform	Editeur	Description
Android	Google	Android est un système d'exploitation développé par Google Inc. et publié en Septembre 2008, suivie par le lancement d'Android Market en Octobre 2008 sont disponibles fournisseur libre de logiciels open source pour tout appareil intelligent.
iOS	Apple	iOS est un système d'exploitation mobile développé par Apple Inc., d'abord été dévoilée en 2007 pour l'iPhone d'Apple, et distribué exclusivement au matériel Apple. Il a la deuxième plus grande base installée de smartphones Android plus tard. Il est fermé et la source exclusive, construite sur l'open source Darwin OS noyau.
Windows Phone	MicroSoft	Windows Phone est Microsoft, qui a été lancé en Octobre 2010 avec sa propre boutique d'applications. Il est développé avec le code fermé et propriétaire. Il a la troisième plus grande base installée de smartphones après Android et iOS.
RIM Blackberry OS	BlackBerry	BlackBerry 10 (basé sur QNX OS) est le BlackBerry. Comme téléphone intelligent est fermé et code propriétaire. BlackBerry 10 est la plate-forme de prochaine génération pour smartphones et tablettes BlackBerry, dont l'App Store a été lancé en Avril 2009. Tous les téléphones et les tablettes sont fabriqués par BlackBerry. Depuis, il est l'une des plates-formes dominantes dans le monde, sa part globale du marché a été réduit à moins de 1% en fin 2014.

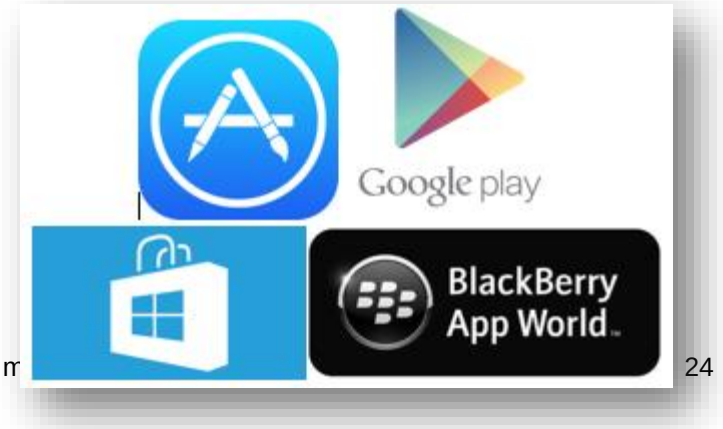
Applications mobiles



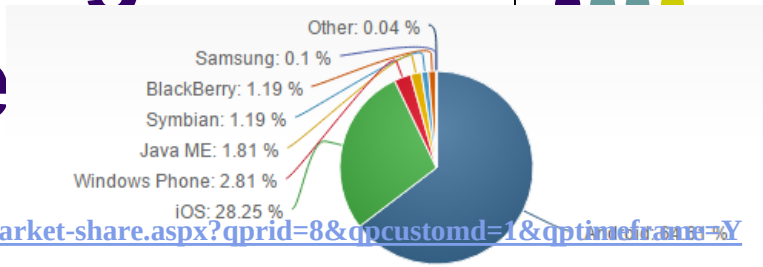
- Une application mobile est un logiciel applicatif développé pour un appareil électronique mobile, tel qu'un assistant personnel, un téléphone portable, un «smartphone», un baladeur numérique, une tablette tactile, ou encore certains ordinateurs fonctionnant avec le système d'exploitation Windows Phone.
- Elles sont pour la plupart distribuées depuis des plateformes de téléchargement (parfois elles-mêmes contrôlées par les fabricants de smartphones) telles que l'App Store (plateforme d'Apple), le Google Play (plateforme de Google / Android), ou encore le Windows Phone Store (plateforme de Microsoft).

Principales plateformes d'application mobiles en 2016

- En 2009, le nombre de téléchargements d'applications mobiles sont élevées, à environ 2,52 milliards de dollars US et devraient atteindre 268,69 milliards en 2017.
- En 2010, les applications mobiles ont rapportés 6,8 milliards de dollars.

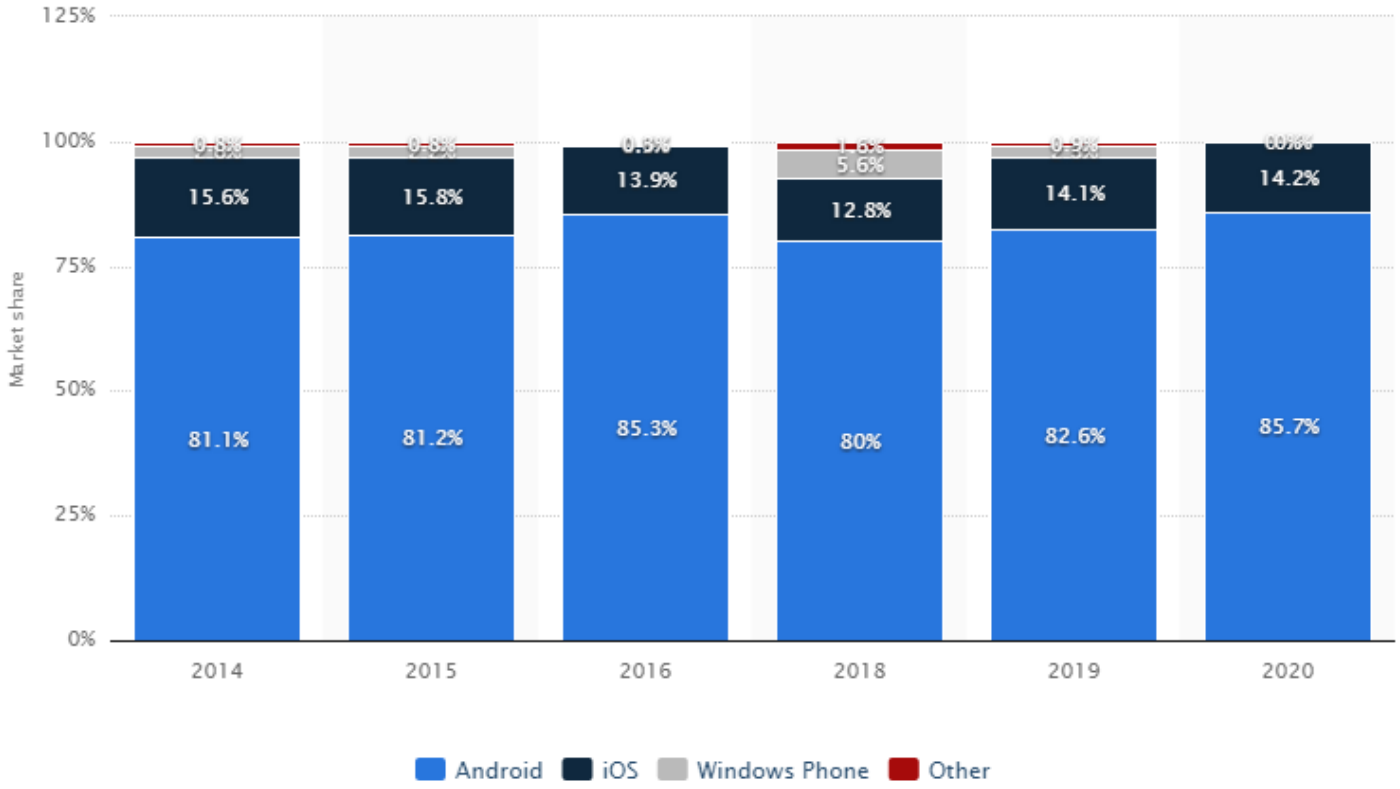


Mobile/Tablet Operating System Market Share



https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qptimeframe=Y

- January, 2016 to September, 2016

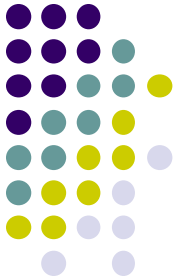


Android ????
Histoire

- OS mobile
- Anecdote
  - La société Android est allée voir en premier Samsung, qui a renvoyé gentiment les représentants de la société Android.
  - Google a par la suite racheté Android, et à lancer son propre OS mobile.
- Selon le site android.com, le système est défini comme : « La plate-forme la plus personnalisable et la plus facile à utiliser qui est installé dans plus d'un milliard dispositifs à travers le monde - des téléphones et les tablettes à des montres, de la télévision, de voitures et d'autres à venir»

Android

https://en.wikipedia.org/wiki/Android_(operating_system)



- In July 2011, Google said that 550,000 Android devices were being activated every day, up from 400,000 per day in May, and more than 100 million devices had been activated with 4.4% growth per week.
- In September 2012, 500 million devices had been activated with 1.3 million activations per day.
- In May 2013, at Google I/O, Sundar Pichai announced that 900 million Android devices had been activated.

Android

- Lancé fin 2008, a rapidement été adopté par de nombreux fabricants et est actuellement la plateforme avec la plus forte croissance dans le monde. Le succès d'Android est principalement dû à sa licence *opensource* mais aussi le fait d'avoir un environnement de développement basé sur le langage Java.
- Lancé initialement par Google, l'évolution d'Android est sous la responsabilité l'**Open Handset Alliance**, un consortium qui compte des dizaines de grandes d'entreprises du secteur tels que Google, Samsung, Intel, HTC, Motorola, etc. Ainsi l'OHA se compose :
  - d'opérateur mobile (Vodafone, Teleponica, Telecom Italia, China Mobile, etc.)
  - de fabricants de téléphone mobiles (Asus, HTC, LG, Motorola, etc.)
  - de fabricants de semi-conducteur (Intel, Nvidia, ARM, etc.)
  - d'éditeurs logiciels (Ebay, Google, PacketVideo, etc.)
  - de distributeurs (Aplix corporation, Borqs, TAT)

Evolution des versions

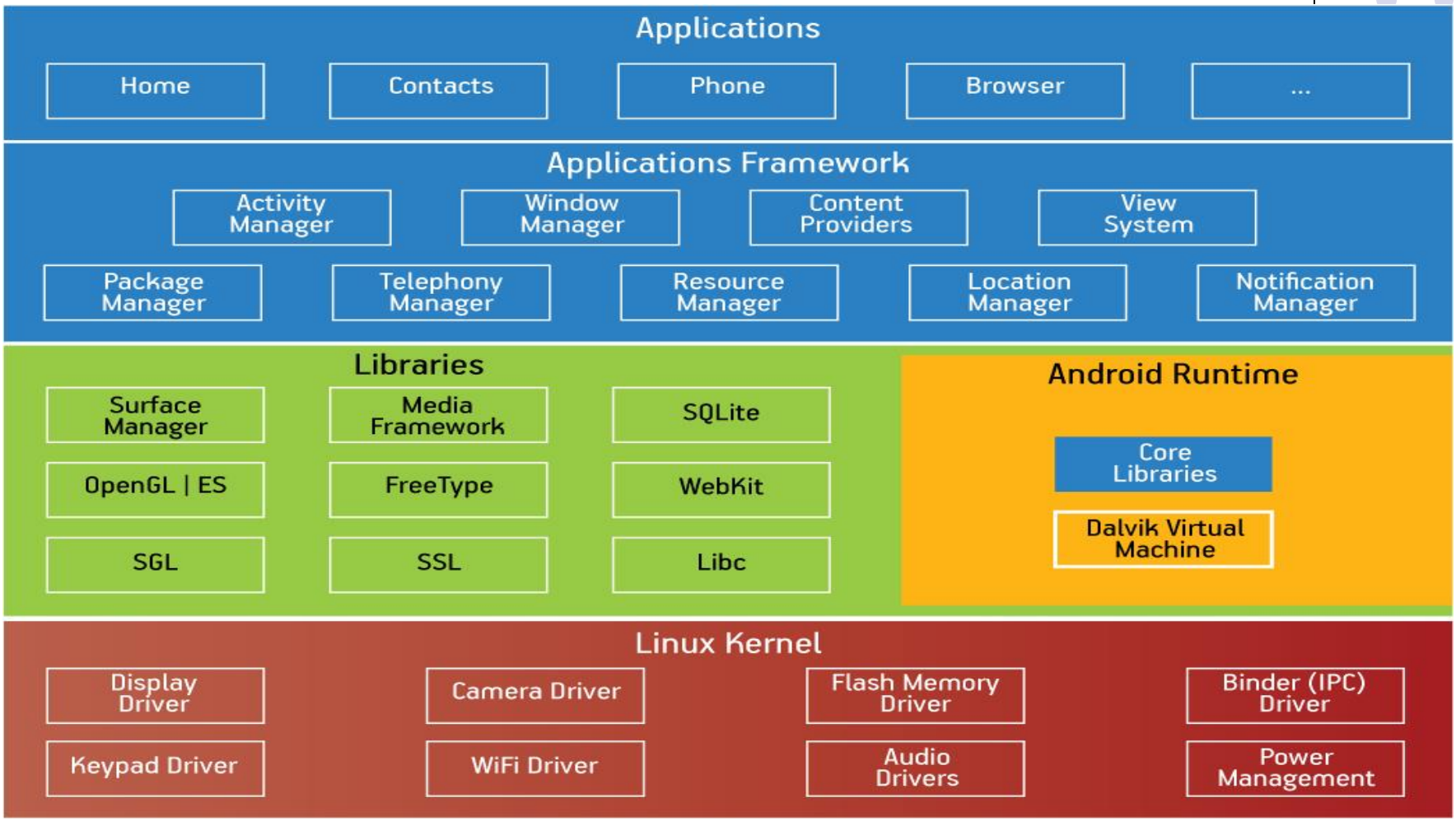
Android's releases are named after sweets or dessert items (except for the first and second releases):

Android framework architecture

Android est un stack logiciel pour terminaux mobiles qui inclut un système d'exploitation, du middleware et des applications clés.

- L'architecture d'Android est divisée en couches:
  - Le noyau Linux (couche inférieure) est la couche qui fournit les principaux services offerts par la plate-forme Android,
  - Les librairies de classe
  - Le runtime d'exécution Android,
  - Le framework Applicatif
  - Les applications, celle que nous apprendrons à développer tout au long de ce cours

L'architecture d'Android

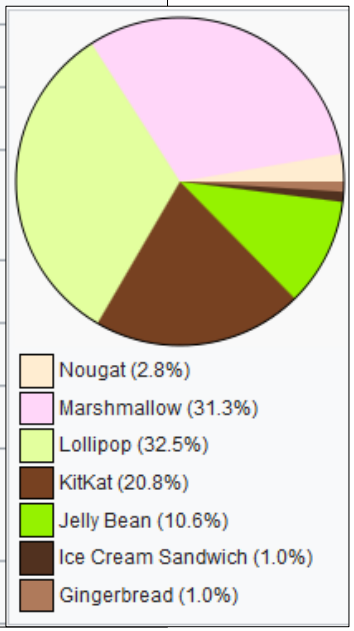


Platform usage

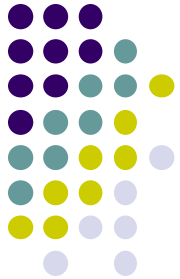
(https://en.wikipedia.org/wiki/Android_(operating_system)

Charts in this section provide breakdowns of Android versions, based on devices accessing the Google Play Store in a seven-day period ending on March 6, 2017.[343][c] Therefore, these statistics exclude devices running various Android forks that do not access the Google Play Store, such as Amazon's Fire tablets.

Version	Code name	Release date	API level	DVM/ART	Distribution	First devices to run version
7.1	Nougat	October 4, 2016	25	ART	0.4%	Pixel, Pixel XL
7.0		August 22, 2016	24	ART	2.4%	Nexus 5X, Nexus 6P
6.0	Marshmallow	October 5, 2015	23	ART	31.3%	
5.1	Lollipop	March 9, 2015	22	ART	23.1%	Android One
5.0		November 3, 2014	21	ART 2.1.0	9.4%	Nexus 6
4.4	KitKat	October 31, 2013	19	DVM (and ART 1.6.0)	20.8%	Nexus 5
4.3	Jelly Bean	July 24, 2013	18	DVM	1.5%	Nexus 7 2013
4.2		November 13, 2012	17	DVM	5.4%	Nexus 4, Nexus 10
4.1		July 9, 2012	16	DVM	3.7%	Nexus 7
4.0	Ice Cream Sandwich	December 16, 2011	15	DVM	1.0%	Galaxy Nexus
2.3	Gingerbread	February 9, 2011	10	DVM 1.4.0	1.0%	Nexus S

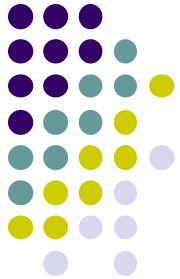


Android: Pourquoi ce choix ?



- Android AOSP
  - Android Open Source Project: Possibilité de modifier Android comme l'on veut. (Nombreuses ROM disponibles)
  - Une large documentation
  - Un store d'application le plus gros existant avec le Play Store
  - Coût financier 25 € à vie (Pour la publication sur le PlayStore) (100 € par an sur IOS)
  - Développement possible depuis n'importe quelle machine: MAC OS X, WINDOWS, LINUX, ANDROID

Comparaison



IOS

- +Application vérifié, et contenant moins de virus que les applications android
- -Prix d'une license (100€/ans)
- -Nécessaire d'avoir un MAC

- IDE : Xcode
- Langage : Objective C, jQuery Mobile et Swift

Android

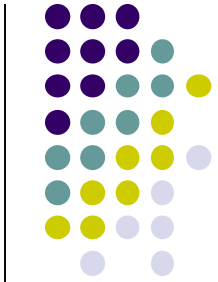
- +Grande communauté d'utilisateurs et de développeur
- +Prix d'une license (25€/vie)
- - Compatibilité non assurés pour les anciennes versions
- IDE : Android Studio / Eclipse / NetBean

- Langage : Java, jQuery Mobile (HTML5/CSS3/JavaScript)

Windows Phone

- - Peu d'utilisateur
- - Compatible uniquement avec marque Lumia
- Ide : Visual Studio
- Langage : C#, jQuery Mobile

IOS : Objectif C et Swift : exemple Hello World

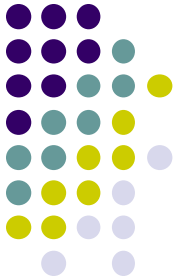


```
#import <Foundation/Foundation.h>
int main (int argc, const char * argv[])
{
        NSAutoreleasePool * pool =
[[NSAutoreleasePool alloc] init];

        // insert code here...
        NSLog(@"Hello, World!");
        [pool drain];
        return 0;
}
```

Java

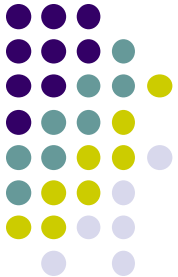
```
Public class main(){
    public static void main(String[] args){
      System.out.print("Hello World !");
    }
}
```

Windows Phone : C#



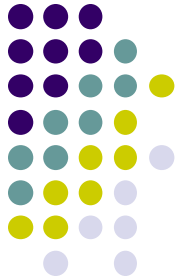
```
using System;
public class HelloWorld
{
    public static int Main()
    {
        Console.Write("Cyril dont have C#!\n");
        Console.ReadKey(); // lets see the display before closing
        return 0;
    }
}
```

All Phones : jQuery phone



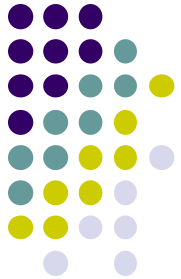
```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css">
<script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
<script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>
</head>
<body>

<div data-role="page">
  <div data-role="header">
    <h1>Bienvenue dans ma page person</h1>
  </div>

  <div data-role="main" class="ui-content">
    <p>Je suis maintenant un développeur Mobile!!</p>
  </div>

  <div data-role="footer">
    <h1>Texte du pied de page</h1>
  </div>
</div>
</body></html>
```

Evaluation



- Définissez un appareil mobile
- Rappelez les principales fonctionnalités des appareils mobiles des décennies 1980-1990,1990-2000 ?
- Quelles sont les évolutions qu'il y a eu durant les années 2000 sur les appareils mobiles ?
- Définir un Système d'exploitation Mobile
- Quels sont les principaux systèmes d'exploitation du Marché

Evaluation



- Définir quelques critères pour choisir une plateforme mobile plutôt qu'une autre
- Comment sont distribuées les applications mobiles ?
- Quelle est l'entreprise qui a créée Android ?
- Sous quelle licence est distribué Android ?
- Quelle est la structure qui gère son évolution?
- Quels sont les principaux éléments de l'architecture d'Android ?

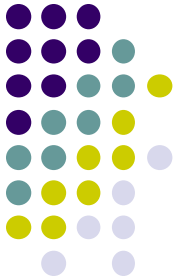
Développement d'applications mobiles

THÉORIES ET PRATIQUE

M2 RT/GLAR

PARTIE 2

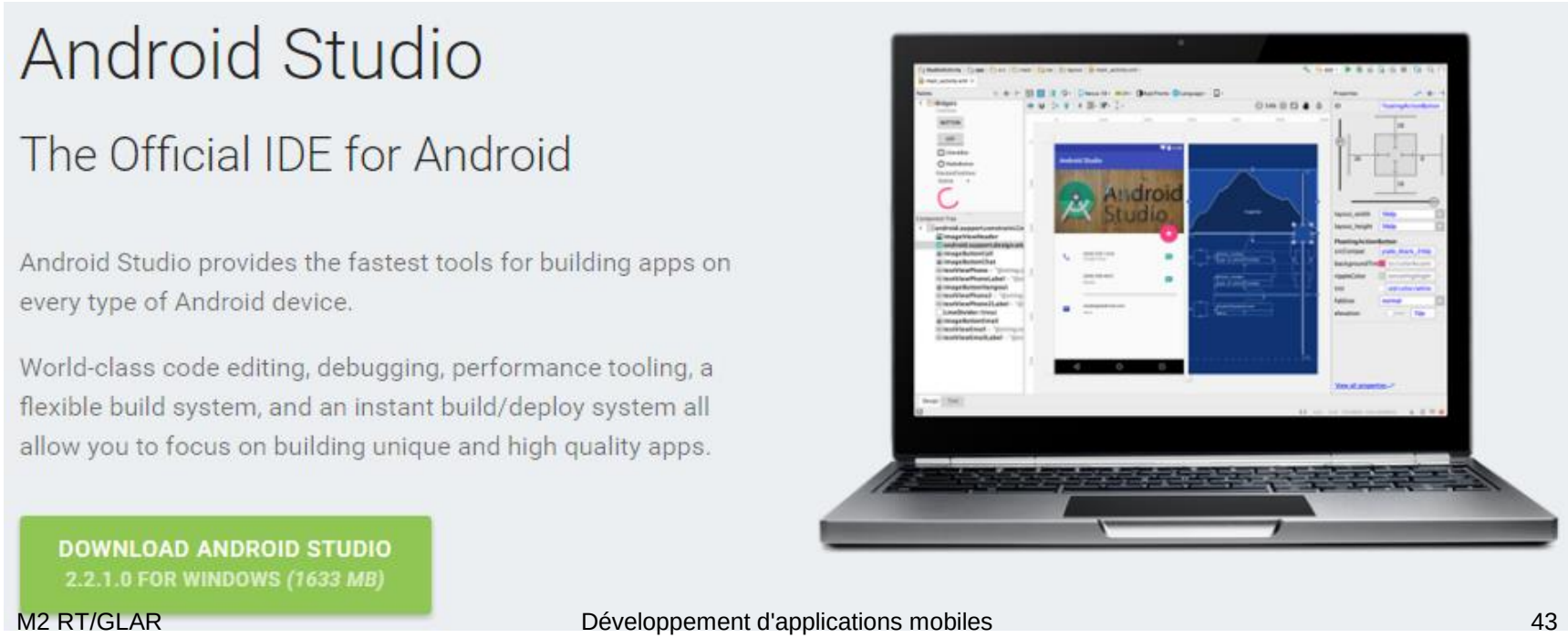
Pr. El hadji M. NGUER



PRÉPARATION DE L’ORDINATEUR

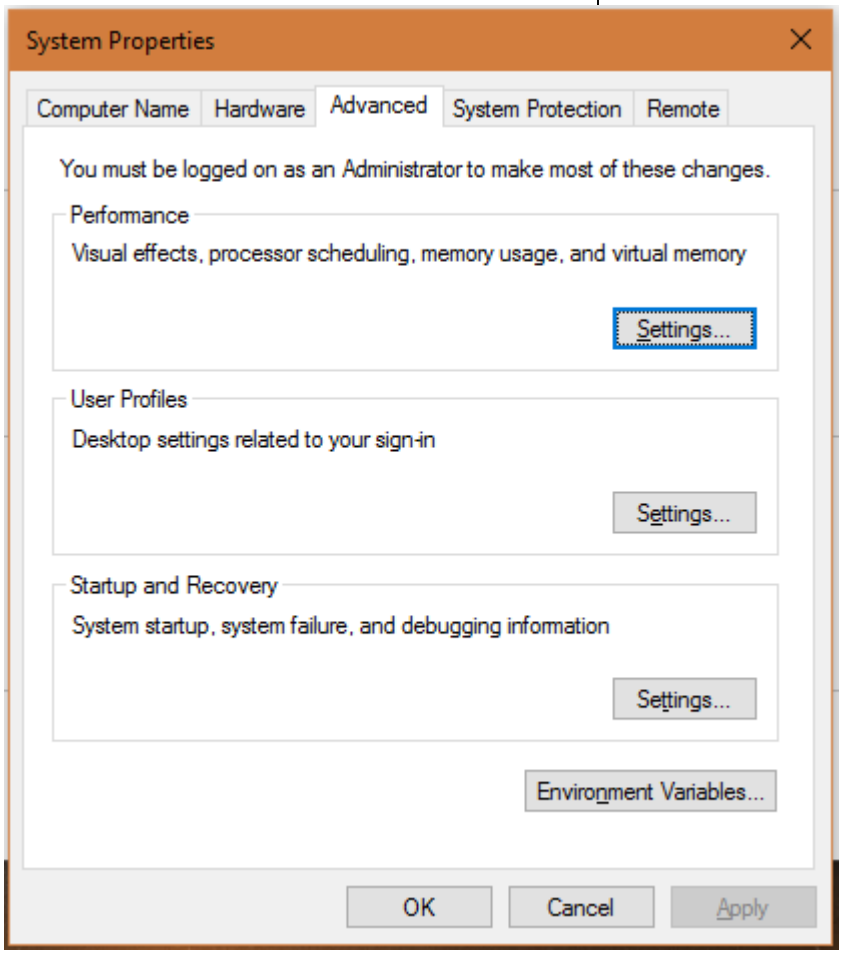
Android - Prerequis

- Java : JDK (Java Developper Kit)
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html

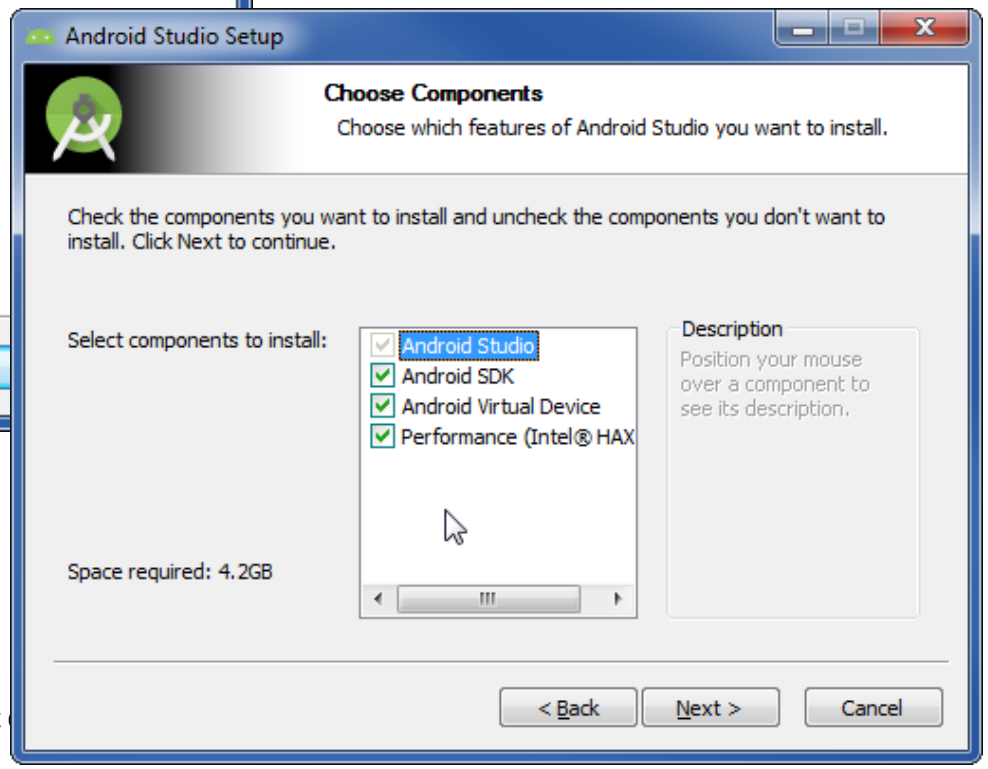


Preparation de l'ordinateur

- Installer Java JDK
  http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html#javasejdk
- Ajouter une variable system
  - Variables d' Environnement
    - Nouveau
      - Nom de la Variable : JDK_HOME
      - Valeur de la variable : Path au repertoire Java (C:\Program Files (x86)\Java)
- Installation Android Studio
  https://developer.android.com/studio/index.html

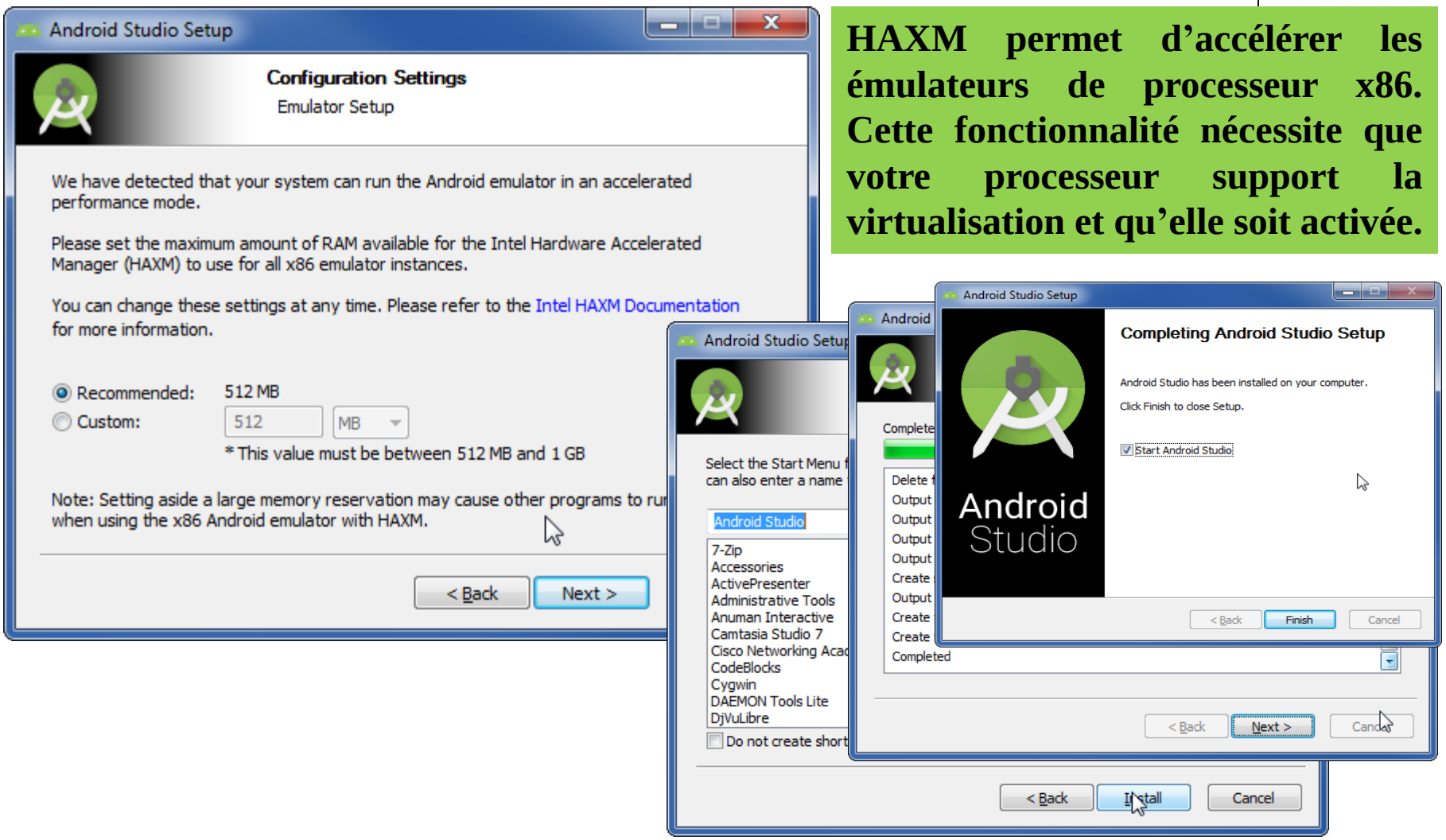


Android Studio Installation



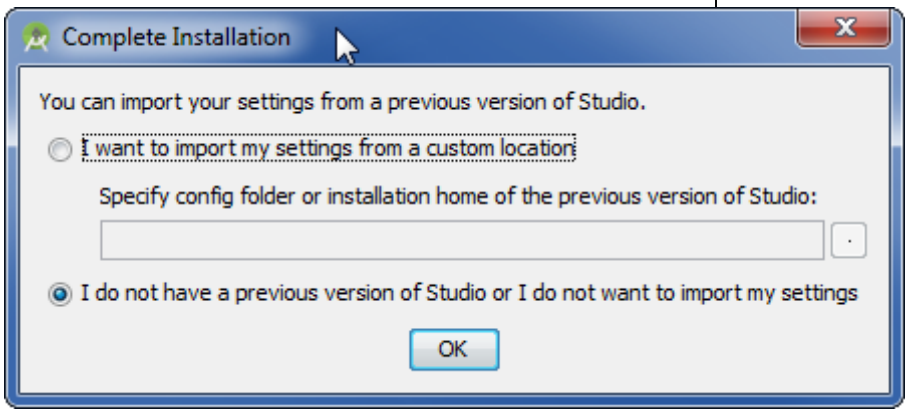
Il faut veiller à ce que le chemin d'installation de Android Studio ait 500MB de libre alors que celui de Android SDK doit avoir au moins 3,2Go de libre.

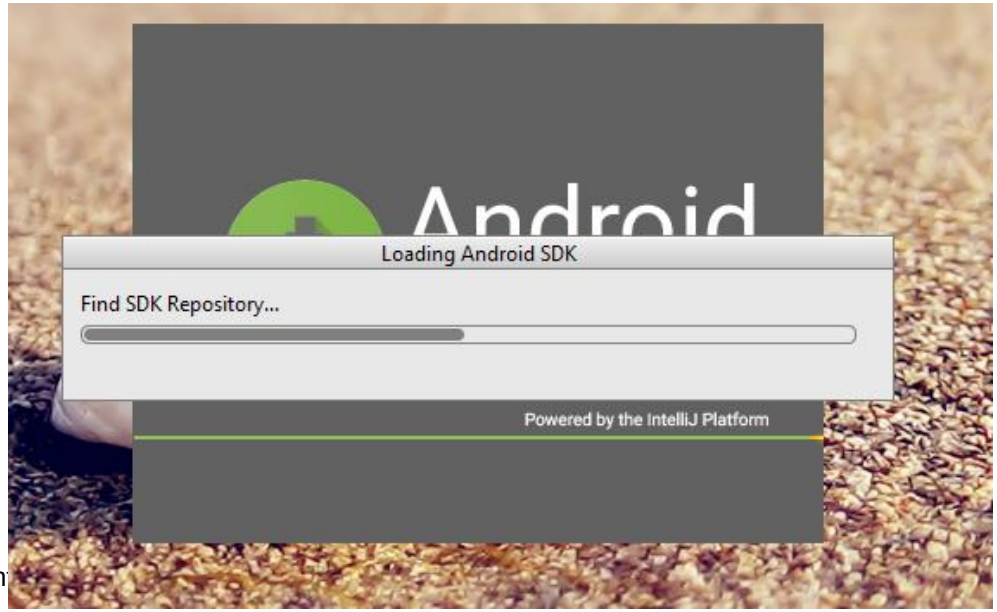
Android Studio Installation



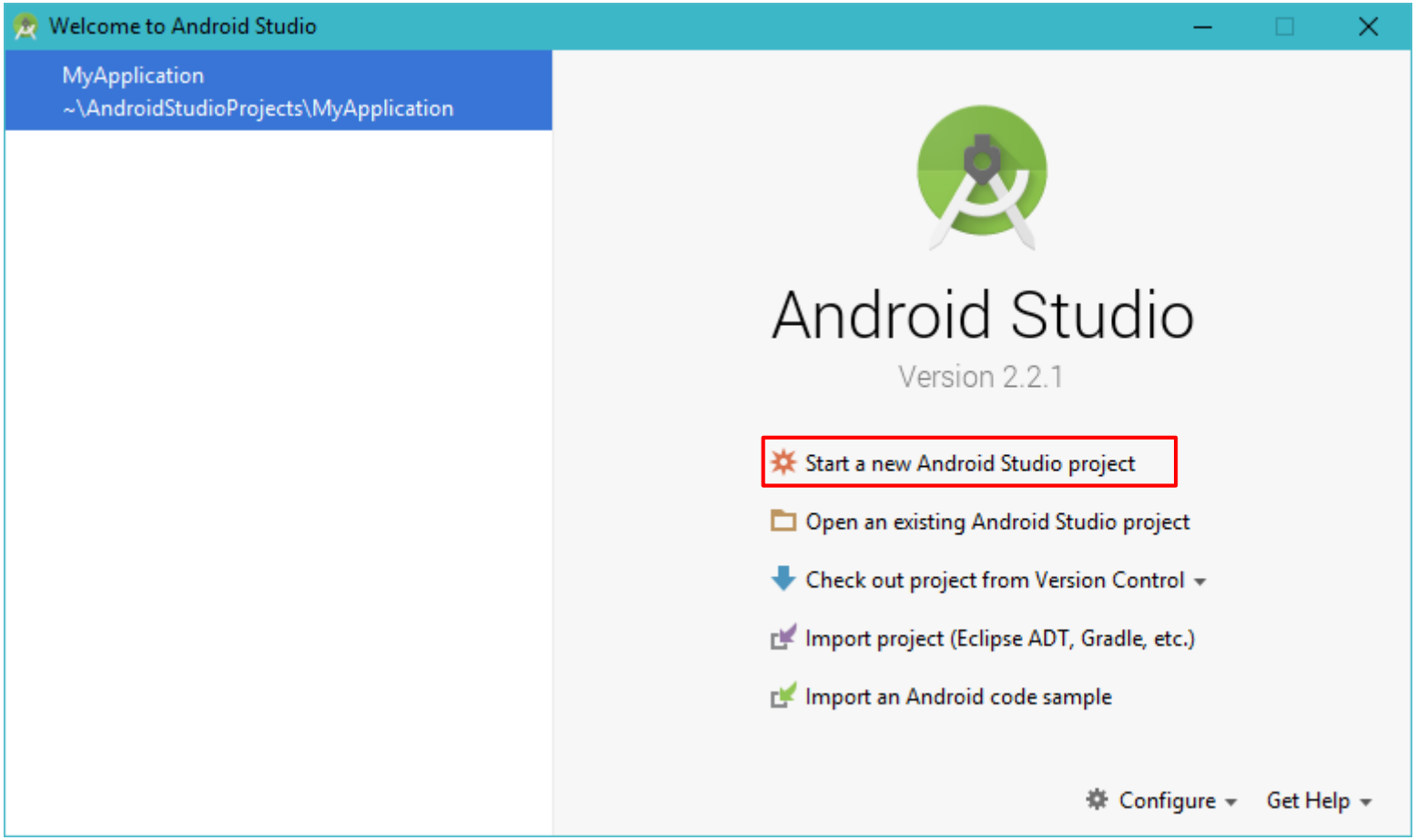
HAXM permet d'accélérer les émulateurs de processeur x86. Cette fonctionnalité nécessite que votre processeur support la virtualisation et qu'elle soit activée.

Lancer Android Studio



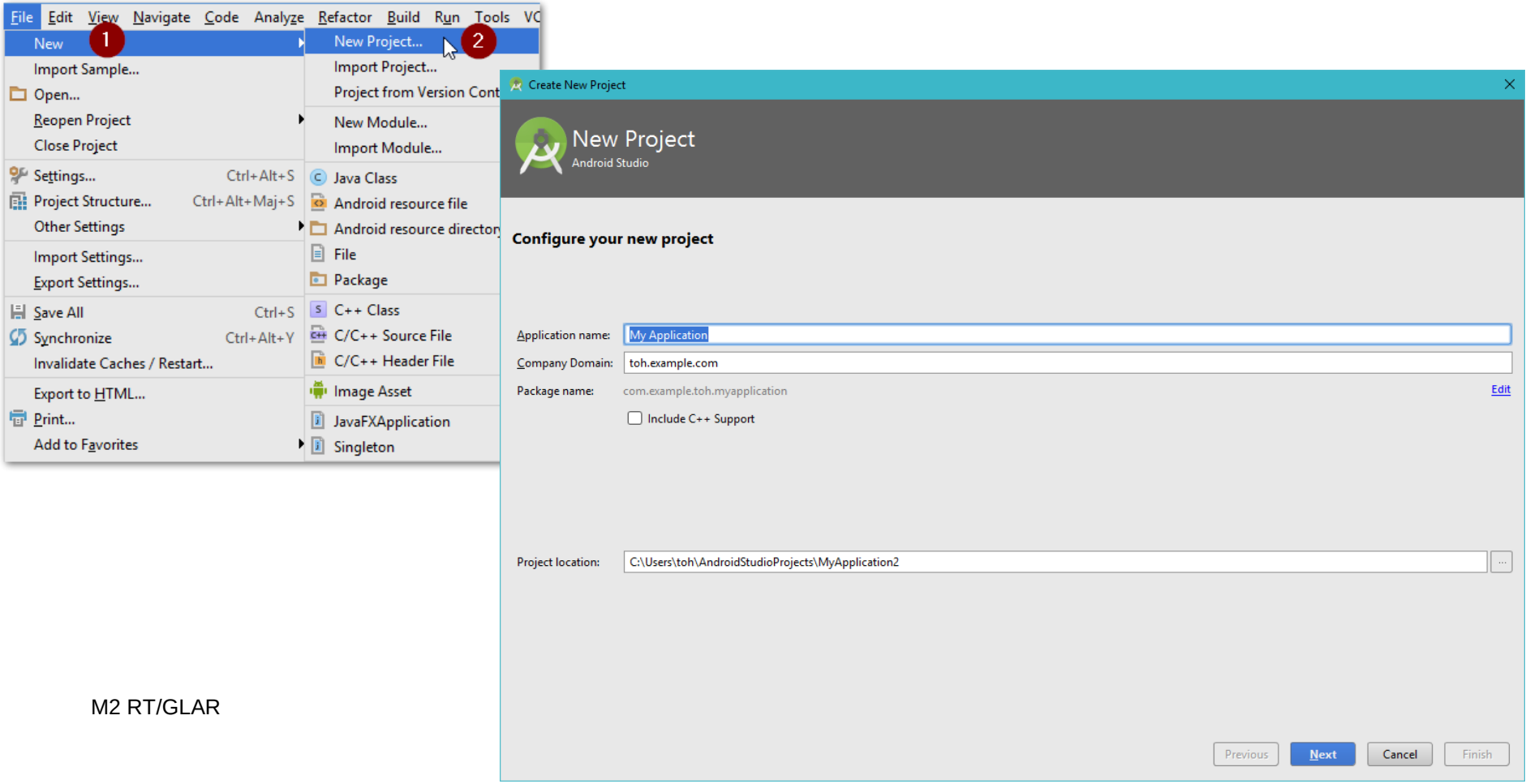


Préparation de votre ordinateur : première possibilité



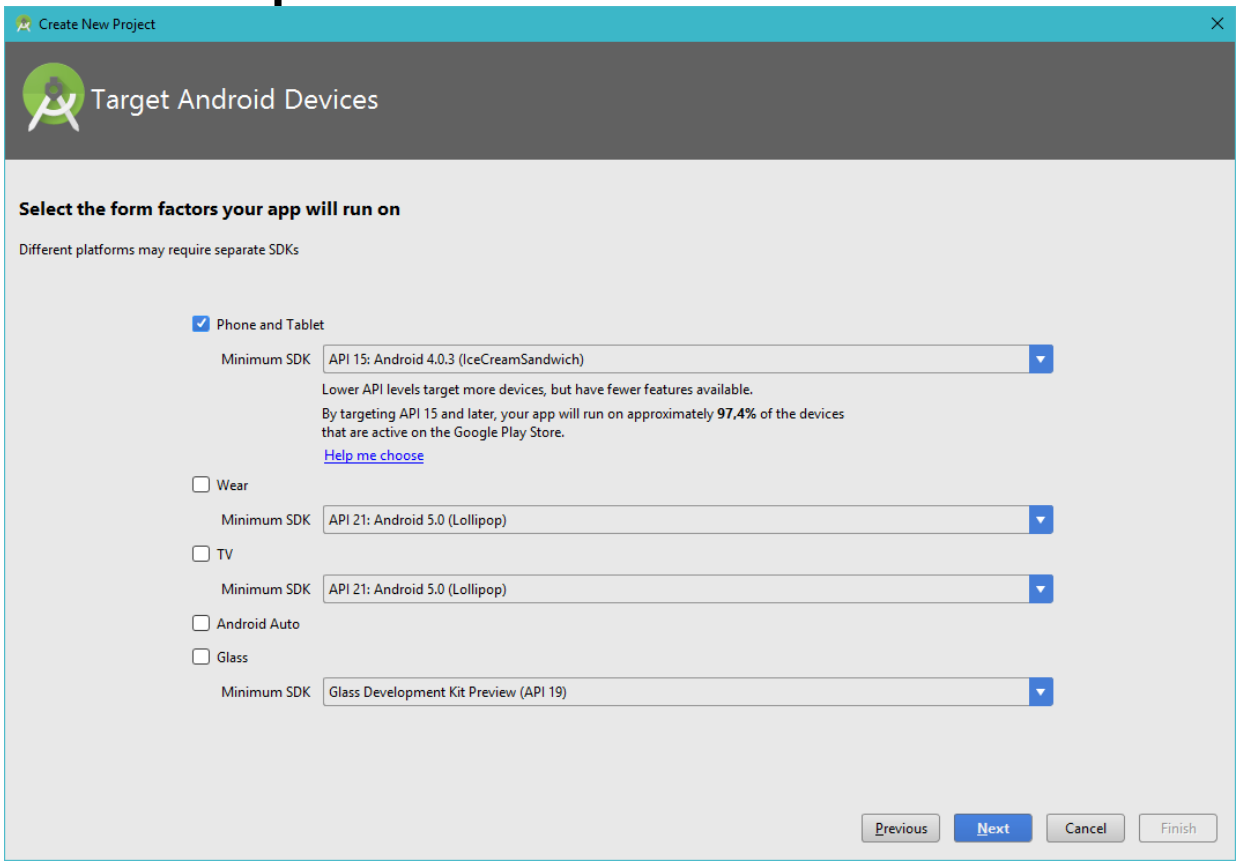
Préparation de votre ordinateur : seconde

- Create a new project : File-> New Project



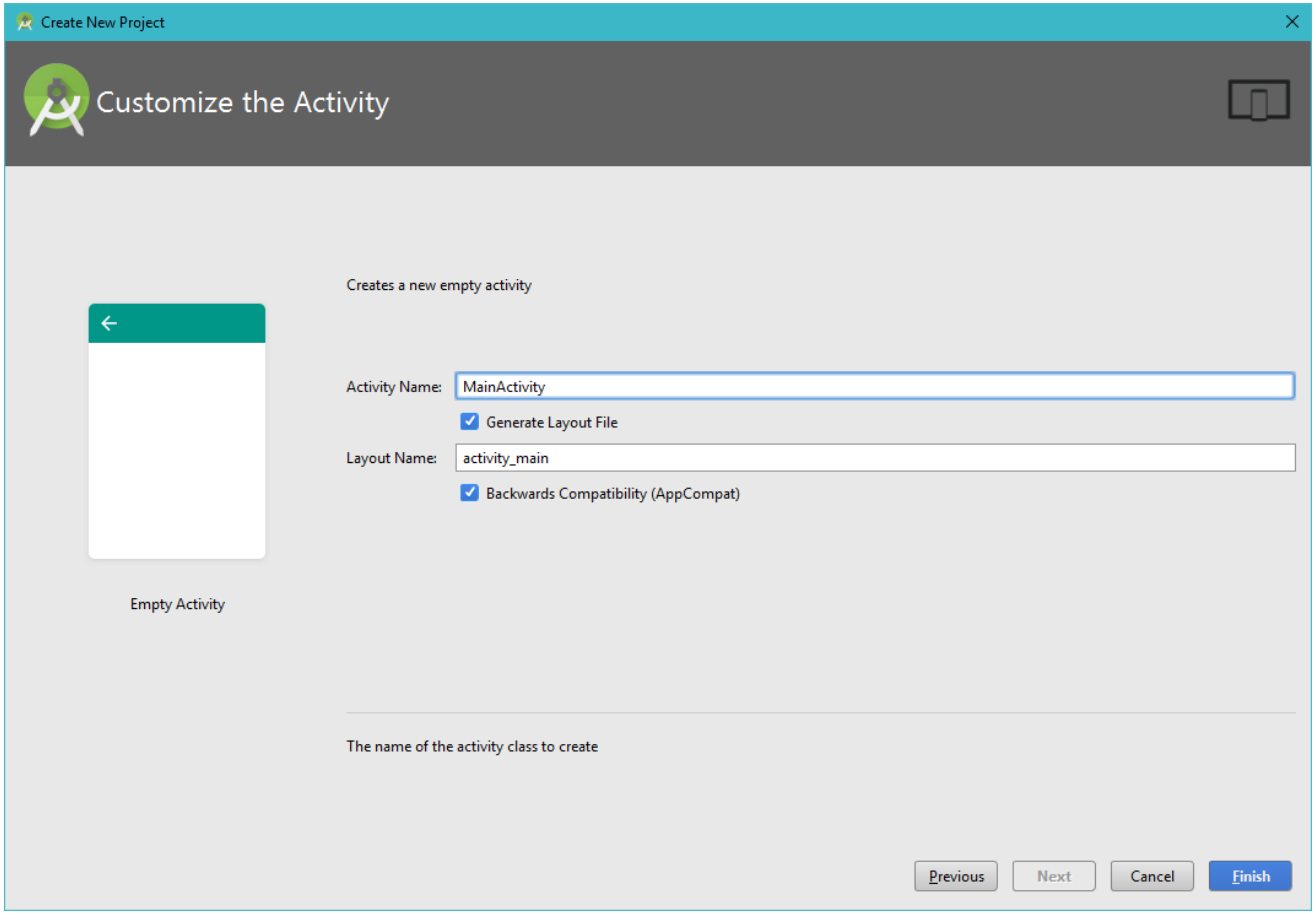
Préparation de votre ordinateur

- Choix de la plateforme



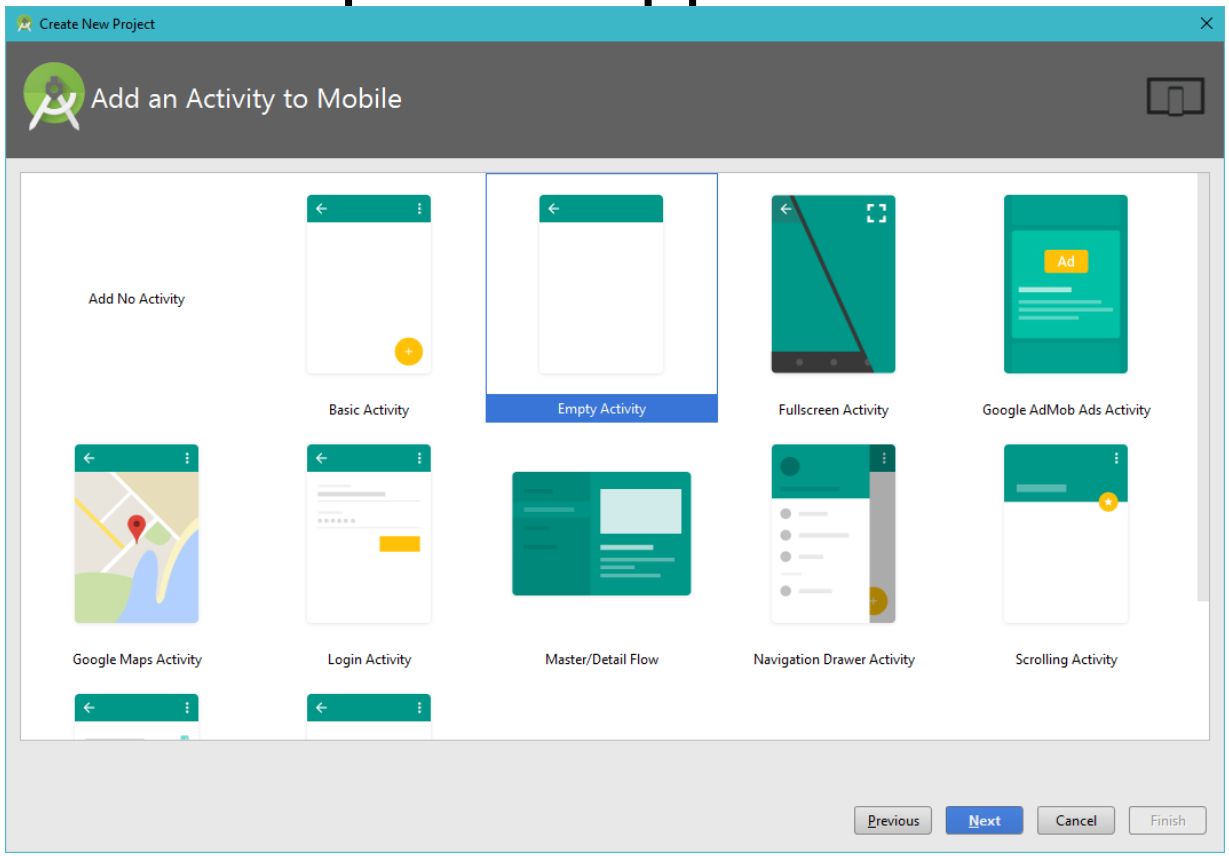
Préparation de votre ordinateur

- Première page de l'assistant de création de projet

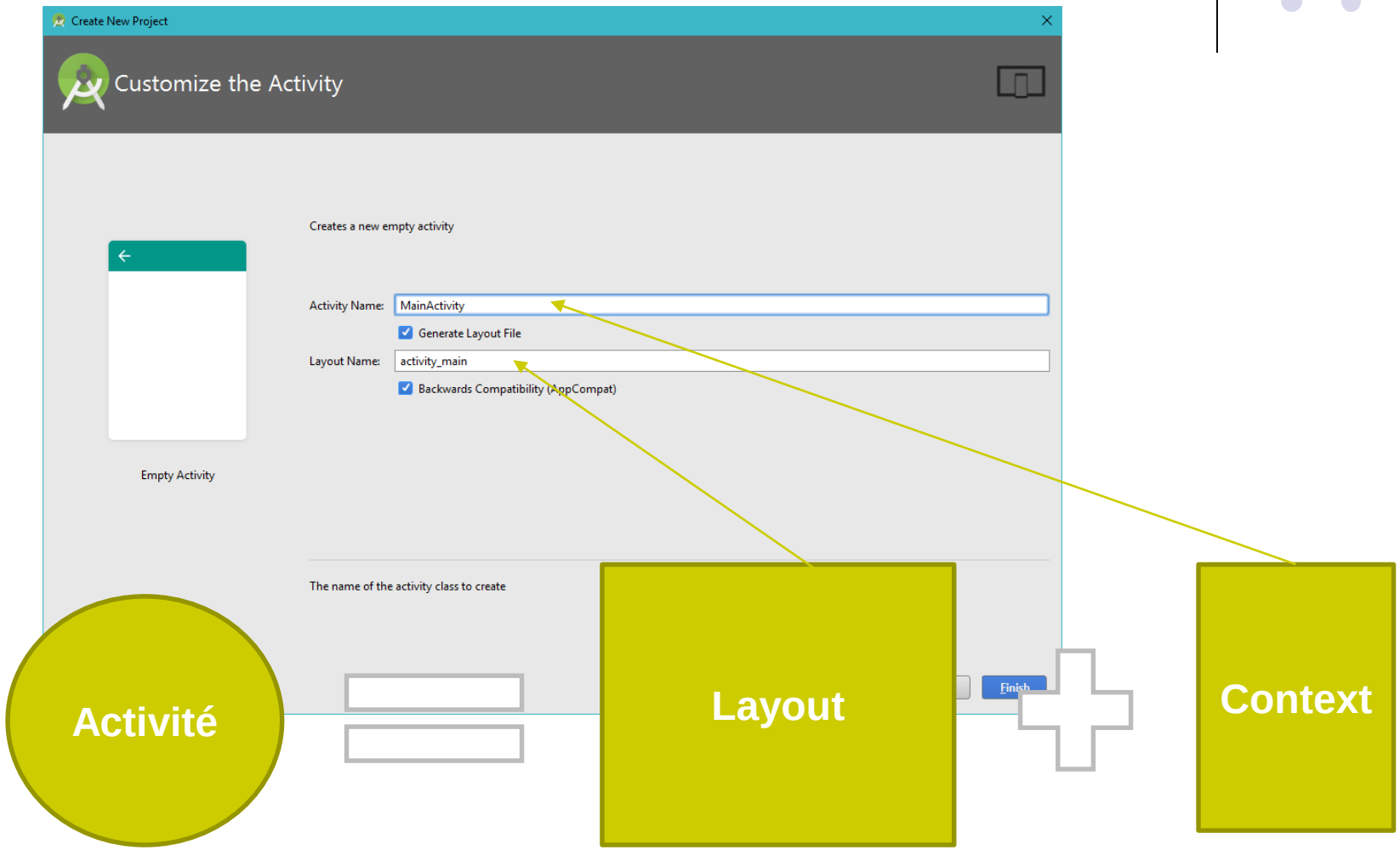


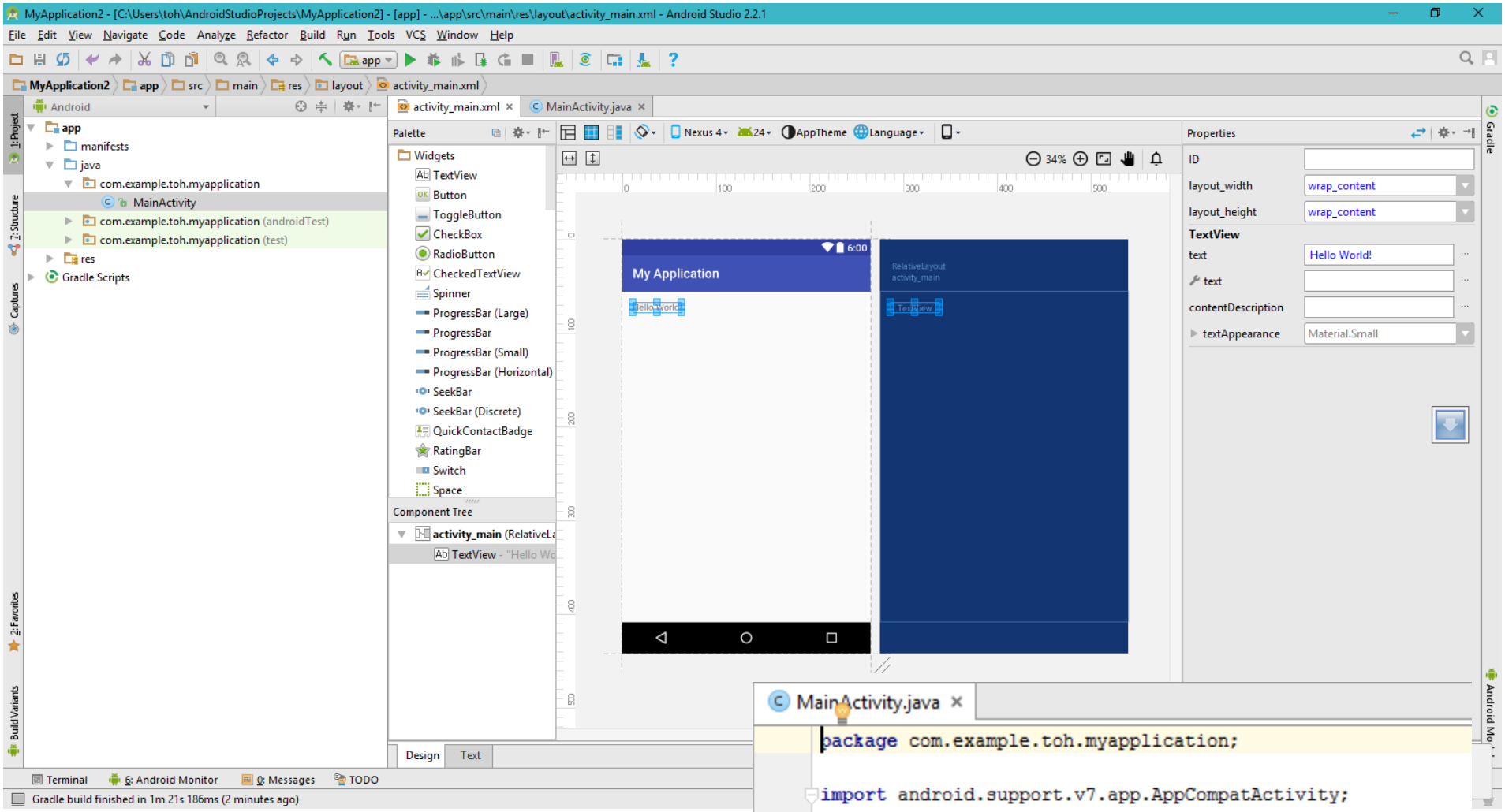
Préparation de votre ordinateur

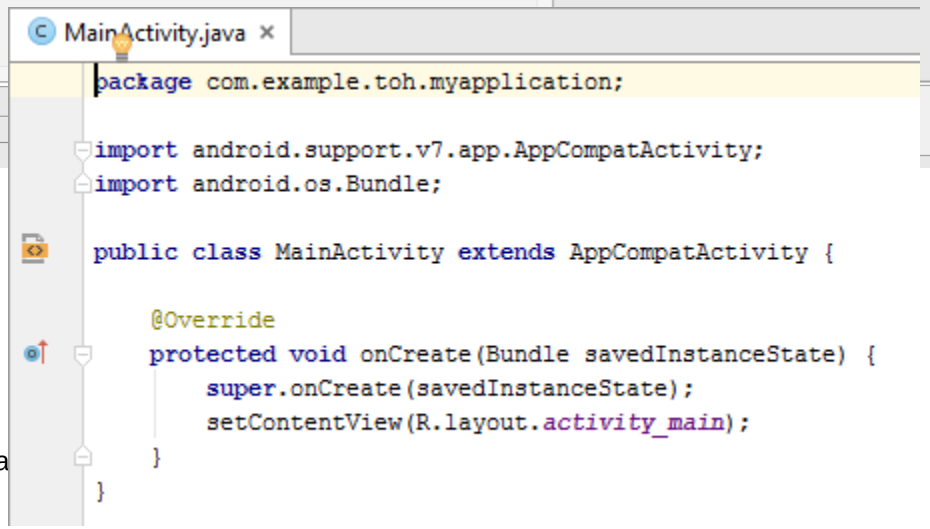
- Choix du Template d'application



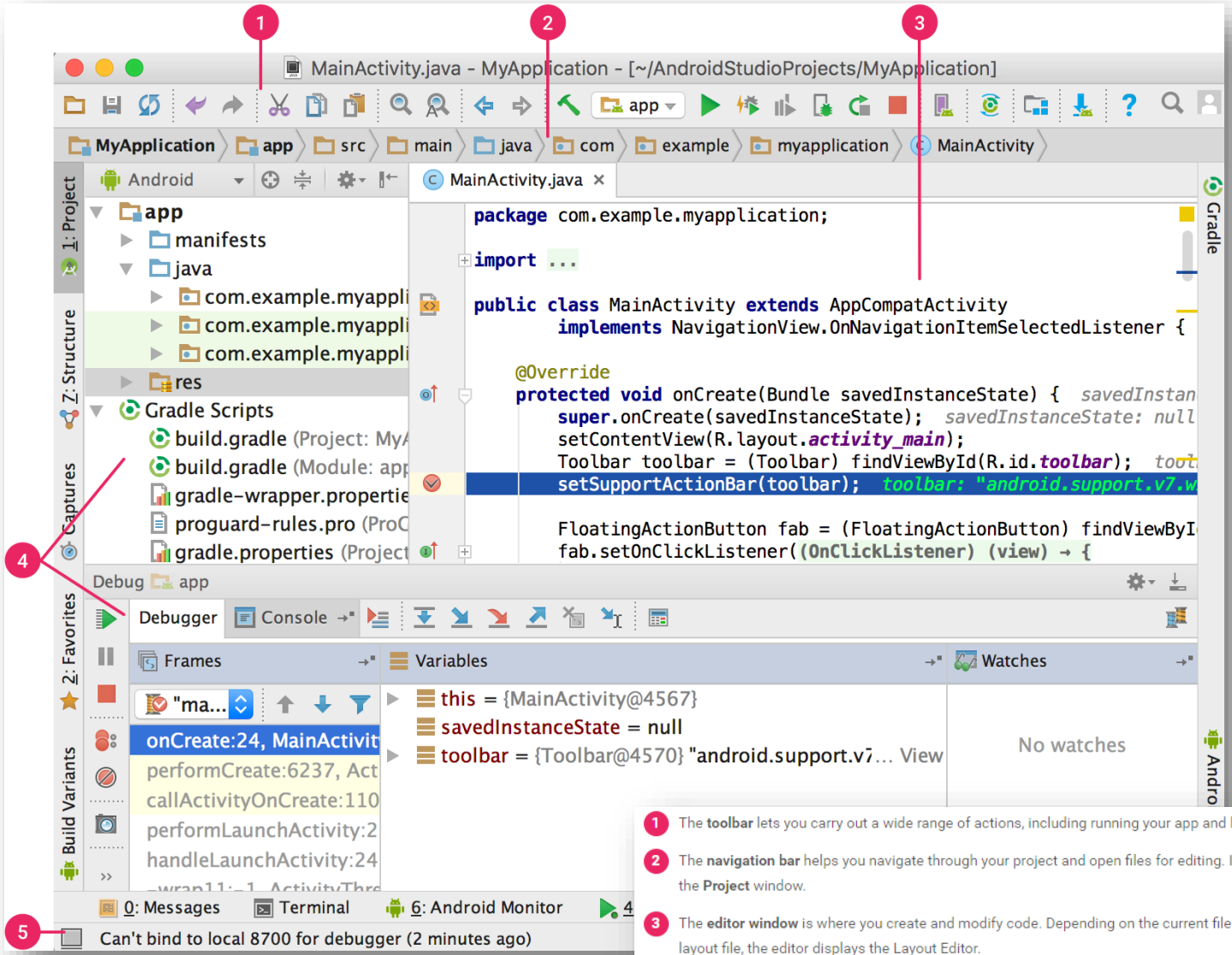
Préparation de votre ordinateur




MyApplication2 - [C:\Users\toh\AndroidStudioProjects\MyApplication2] - [app] - ...\app\src\main\res\layout\activity_main.xml - Android Studio 2.2.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MyApplication2 app src main res layout activity_main.xml
Android
app
manifests
java
com.example.toh.myapplication
MainActivity
com.example.toh.myapplication (androidTest)
com.example.toh.myapplication (test)
res
Gradle Scripts
activity_main.xml
MainActivity.java
Palette
Widgets
TextView
Button
ToggleButton
CheckBox
RadioButton
CheckedTextView
Spinner
ProgressBar (Large)
ProgressBar
ProgressBar (Small)
ProgressBar (Horizontal)
SeekBar
SeekBar (Discrete)
QuickContactBadge
RatingBar
Switch
Space
Component Tree
activity_main (RelativeLa
TextView - "Hello Wo
Nexus 4
AppTheme
Language
34%
My Application
Hello World!
RelativeLayout
activity_main
Properties
ID
layout_width
wrap_content
layout_height
wrap_content
TextView
text
Hello World!
contentDescription
textAppearance
Material.Small
Design
Text
Terminal
6: Android Monitor
0: Messages
TODO
Gradle build finished in 1m 21s 186ms (2 minutes ago)


MainActivity.java
package com.example.toh.myapplication;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
}

Interface utilisateur



1. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
2. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
3. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
4. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
5. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.

Structure du projet

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

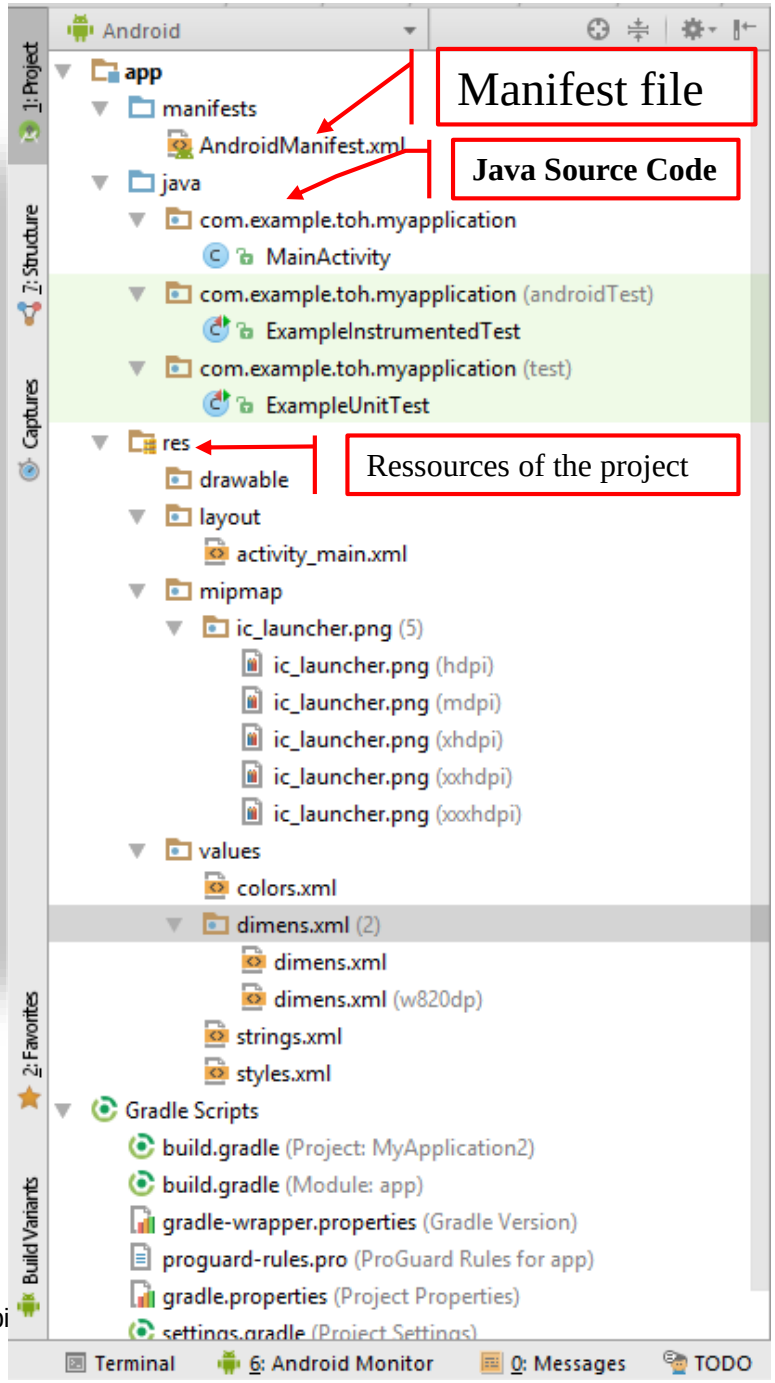
- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

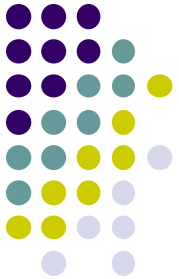
- **manifests**: Contains the `AndroidManifest.xml` file.
- **java**: Contains the Java source code files, including JUnit test code.
- **res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as **Android**).



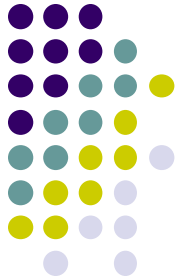
The project consists of three main categories of files:

- The configuration files are XML-like Manifest.xml, activity_main.xml, string.xml ...;
- The Java source files that implement business services, ...
- The resources that the icons and other resources



Felicitations: Vous venez de créer votre premiére application. Lets run it !!!

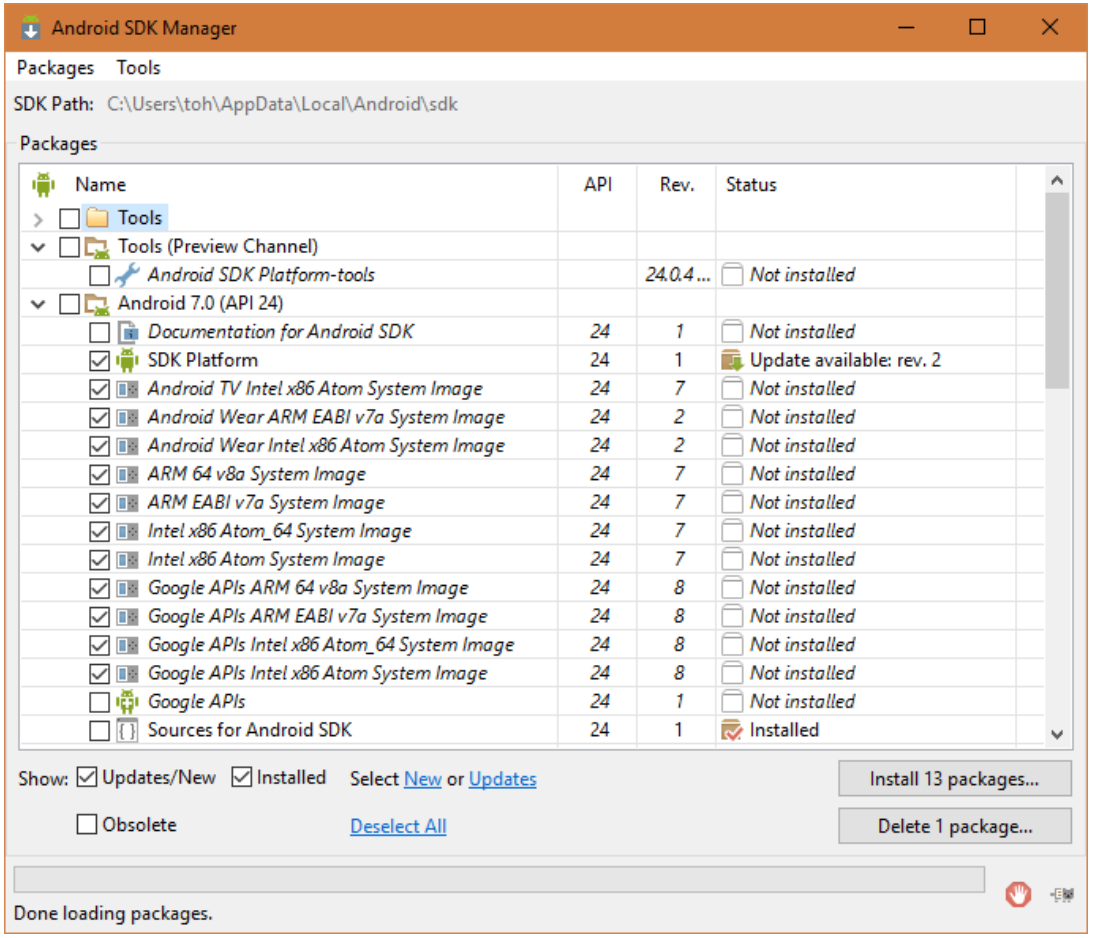
Configuration d’un émulateur



- When installing Android Studio thereof to may be already created an emulator.
- Voici comment en créer un:
  - Choisir la version Android
  - Create a un émulateur
  - Lancer l’émulateur pour les tests

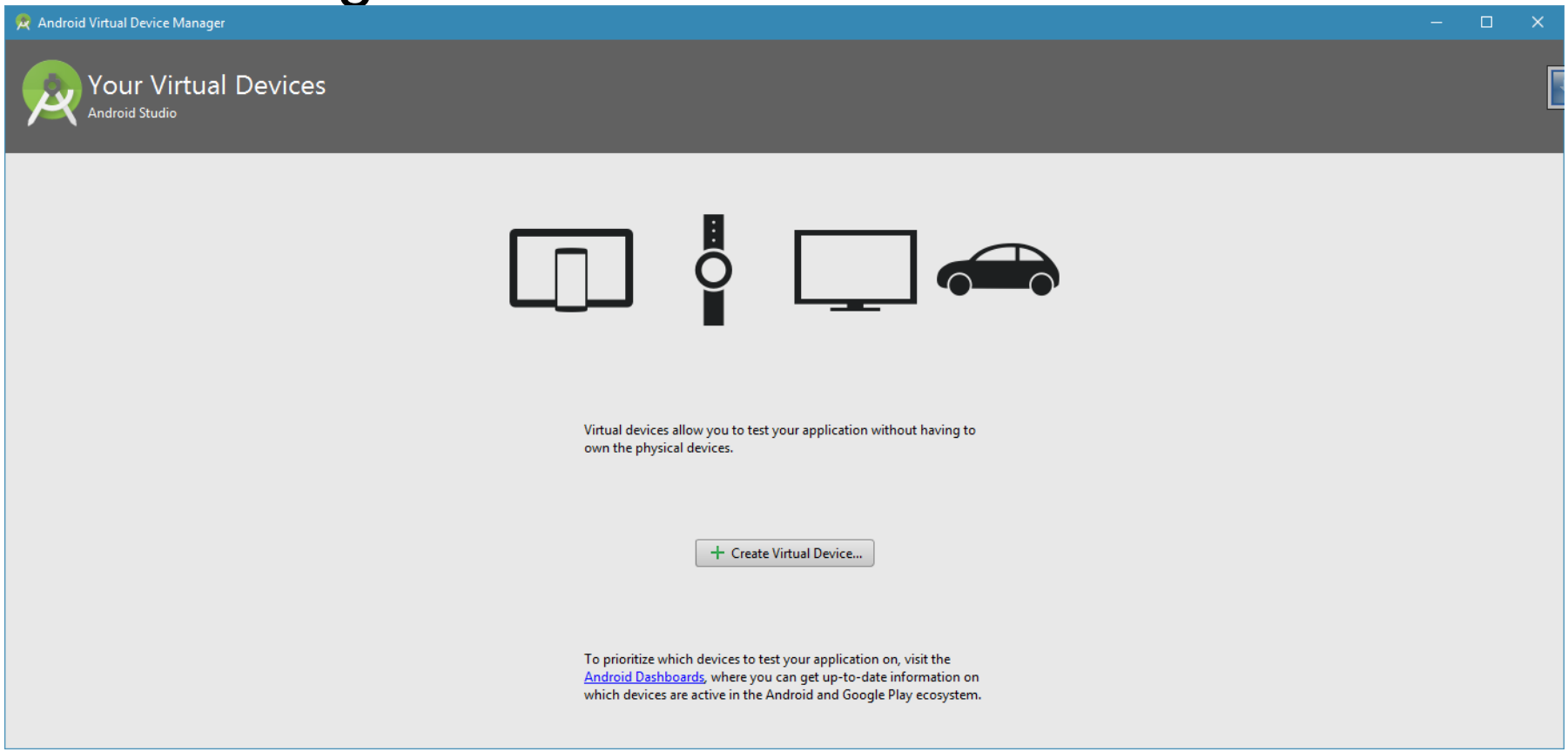
Configuration d'un émulateur

- Choix de la version:
  - Launch SDK (Software Development Kit) Manager



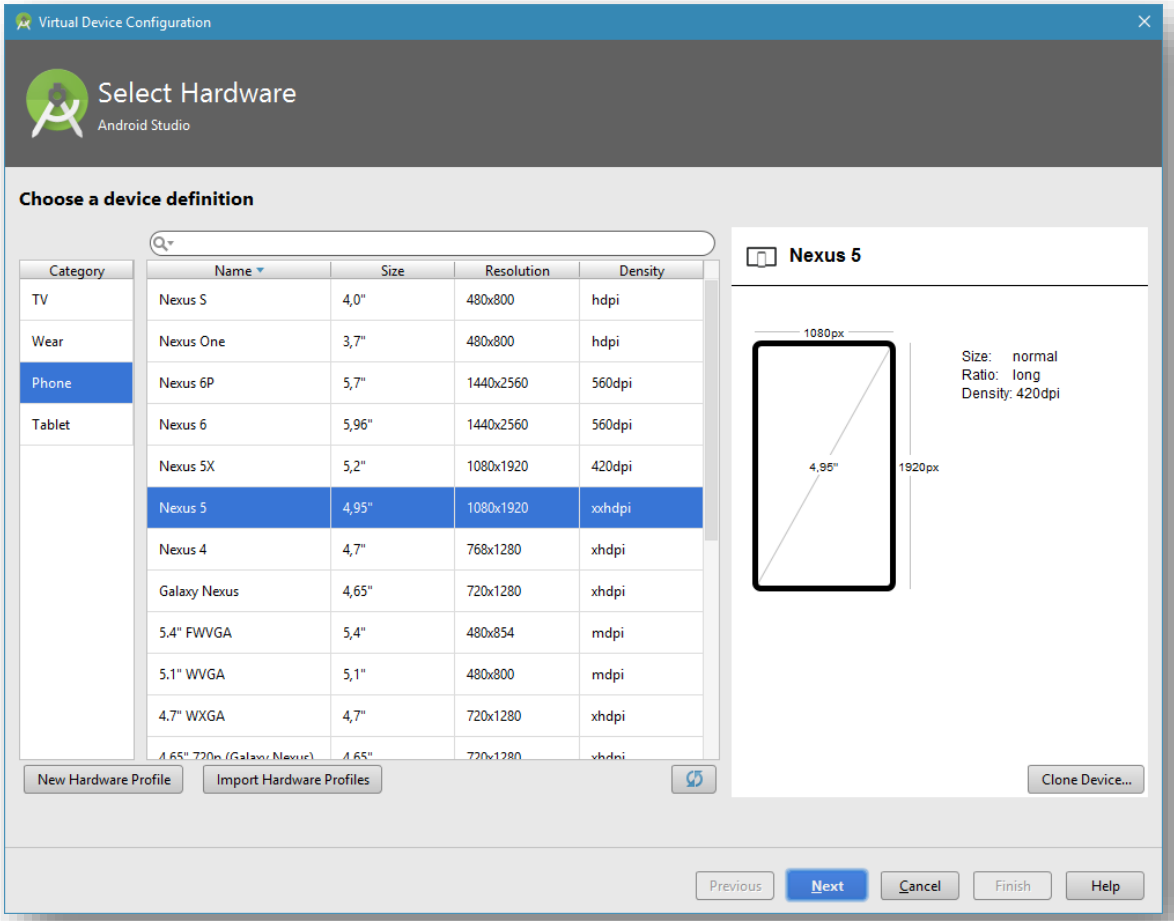
Configuration d’un émulateur

- Creating the Virtual Machine 



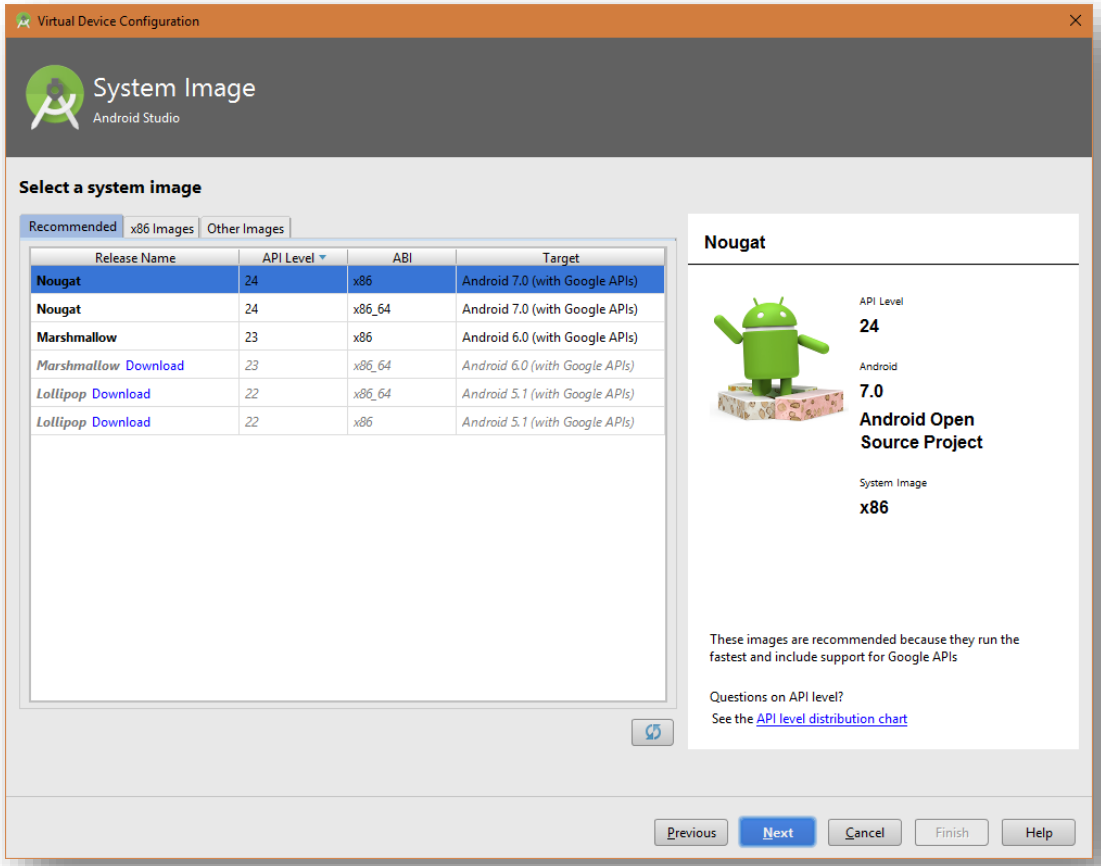
Configuration d’un émulateur

- Creating the Virtual Machine
  - HardWare Choice



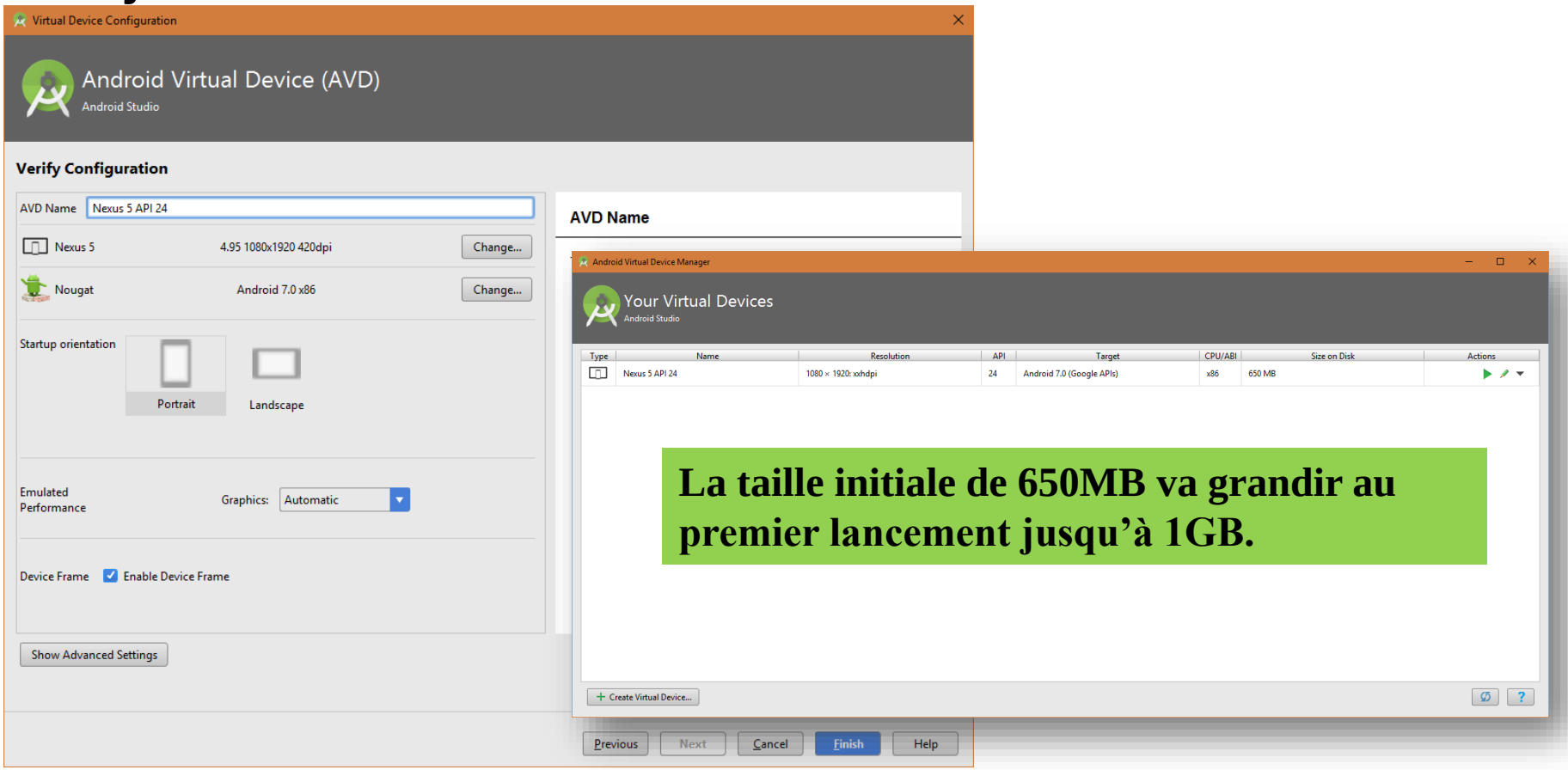
Configuration d'un émulateur

- Creating the Virtual Machine
  - HardWare Choice



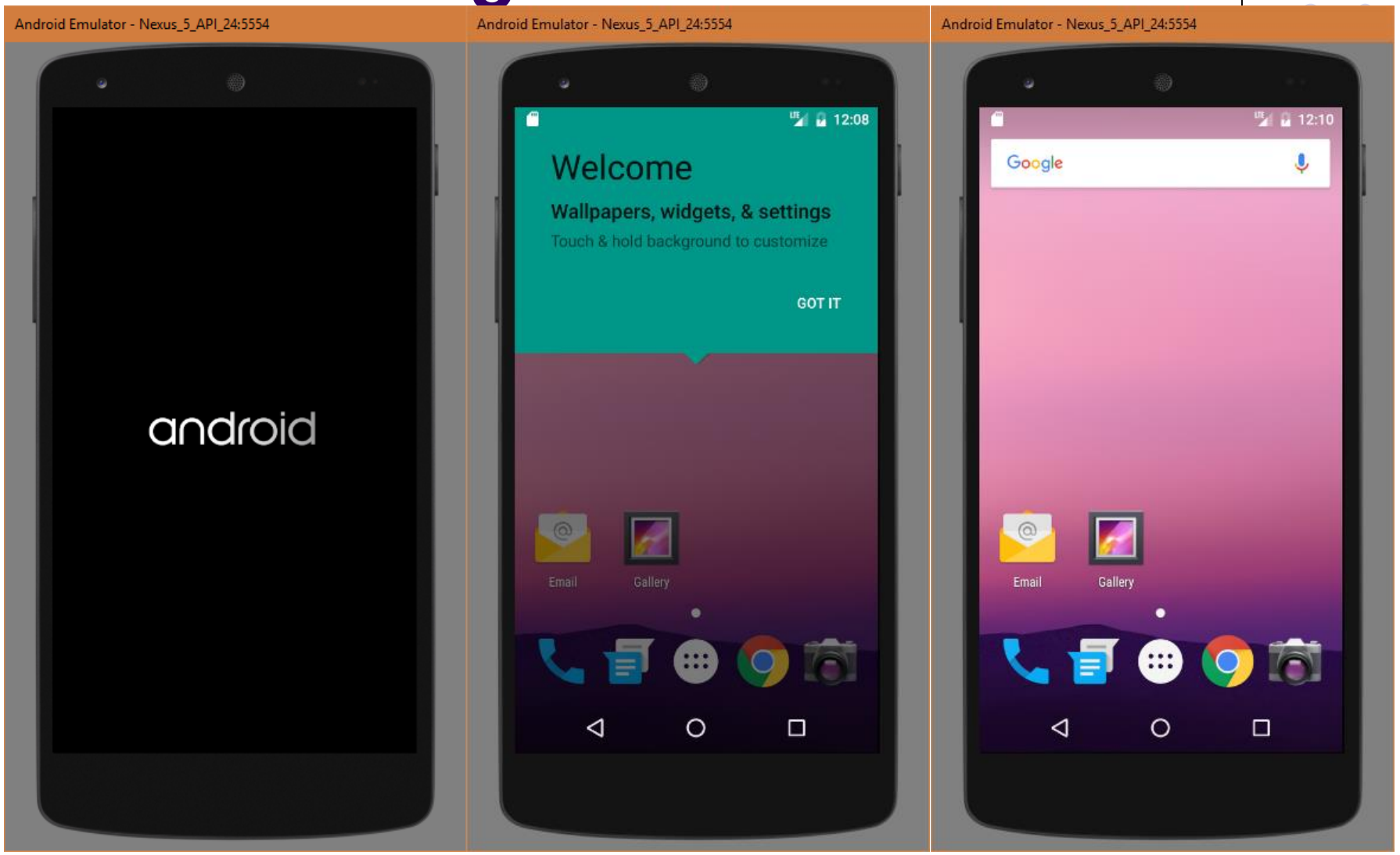
Configuration d’un émulateur

- Ajustements finaux

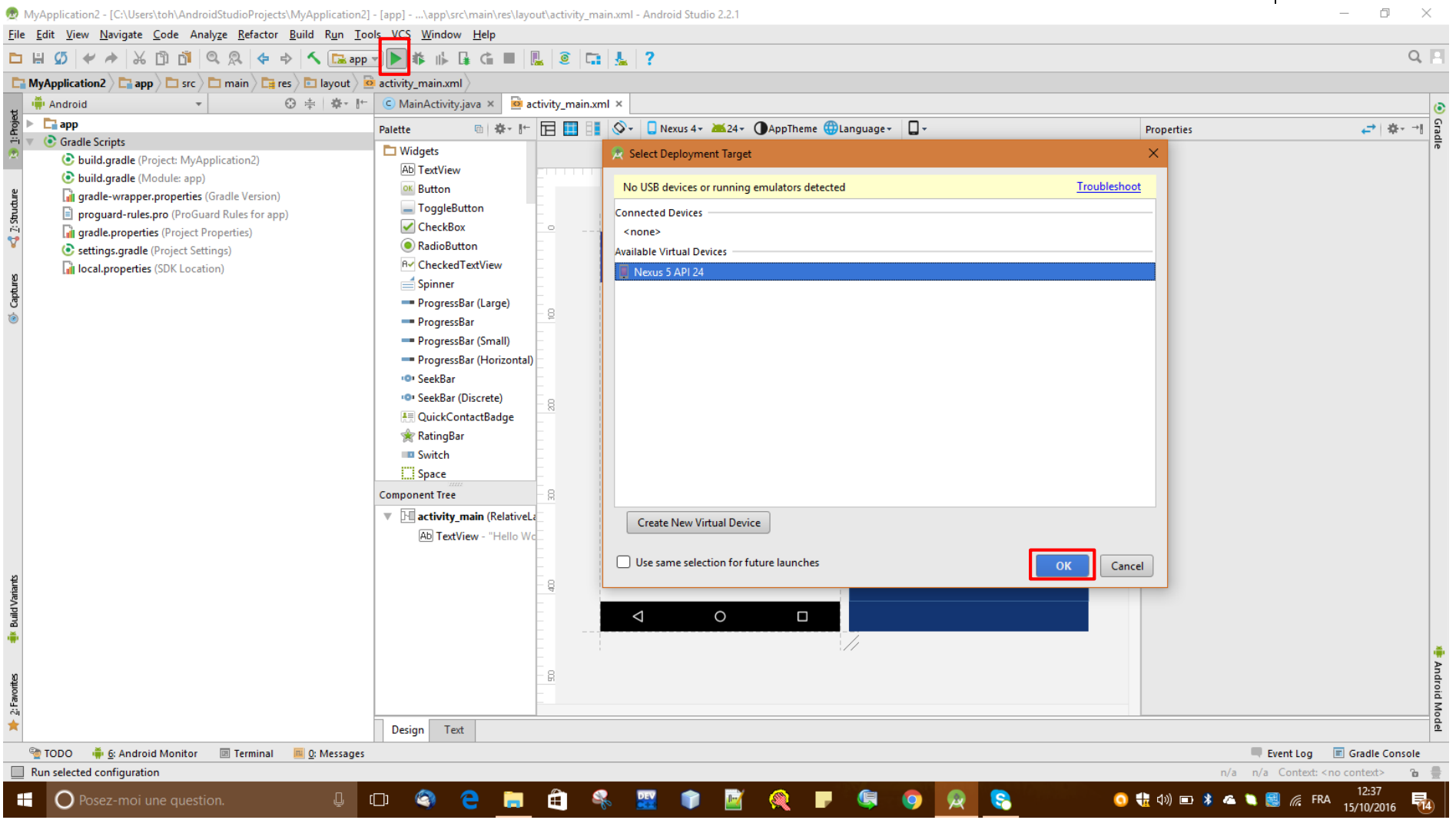


La taille initiale de 650MB va grandir au premier lancement jusqu’à 1GB.

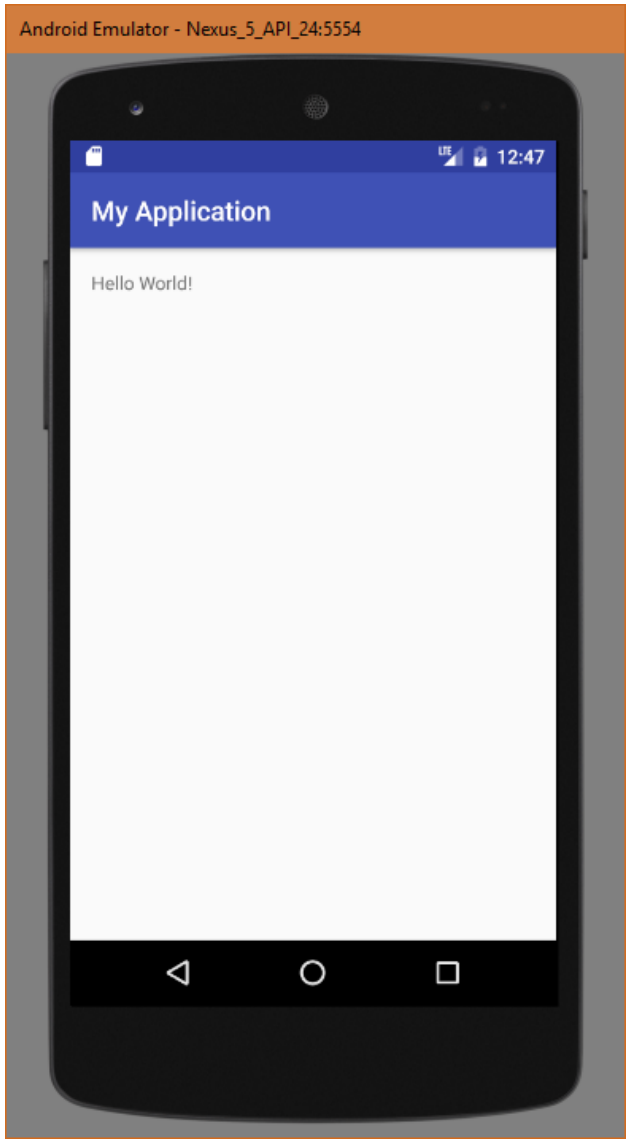
Capture des différentes étapes du démarrage de l'emulateur

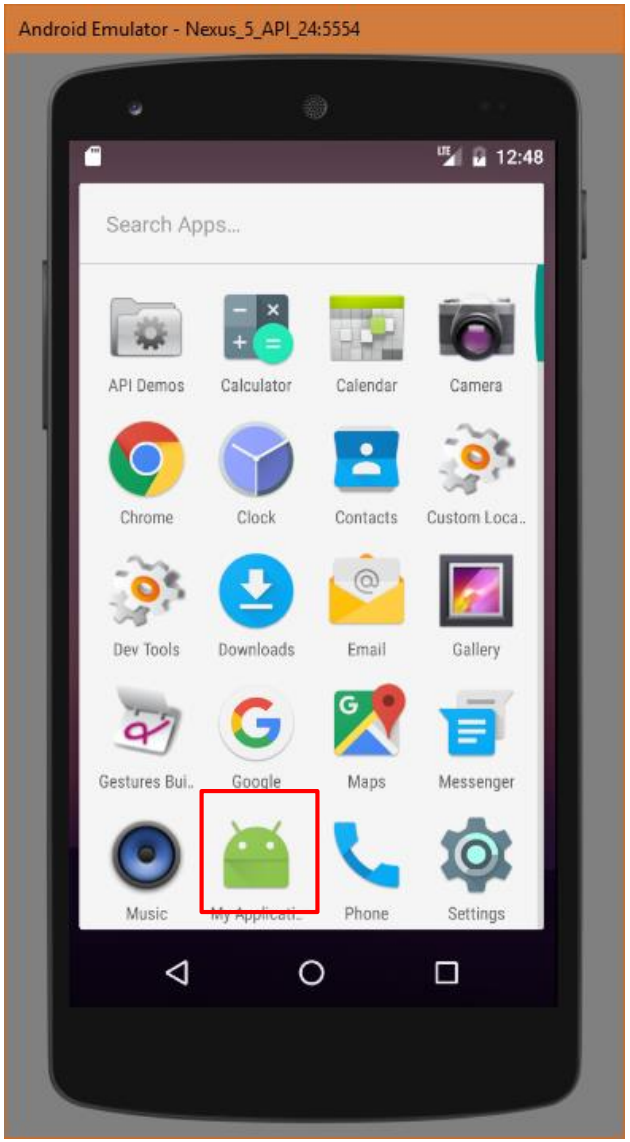


Exécuter votre Application

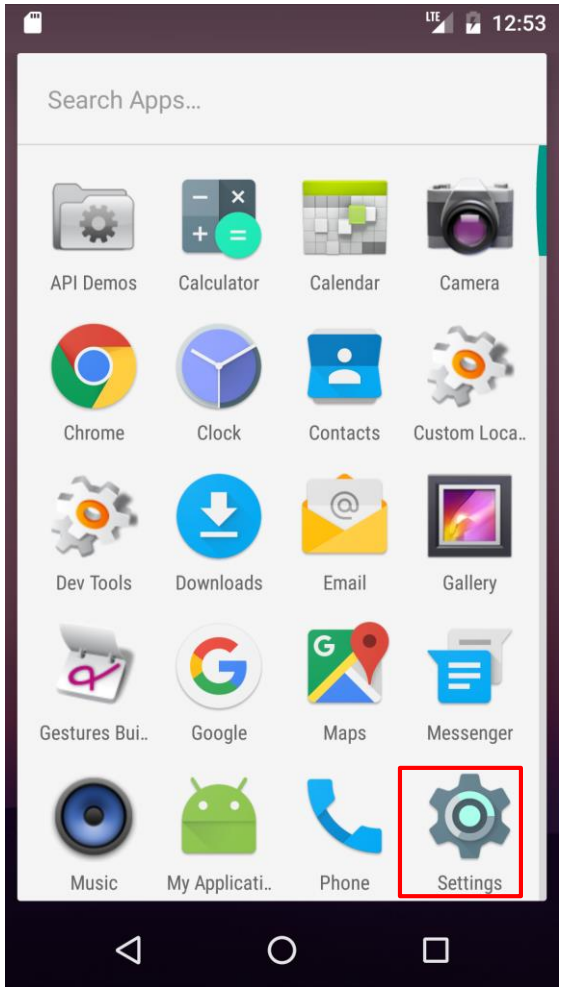


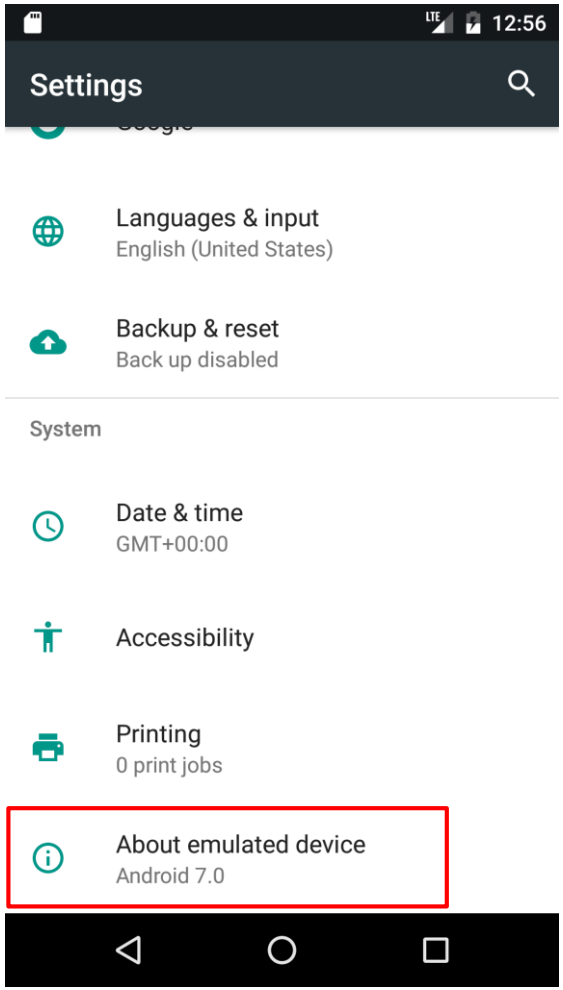
Exécuter votre Application

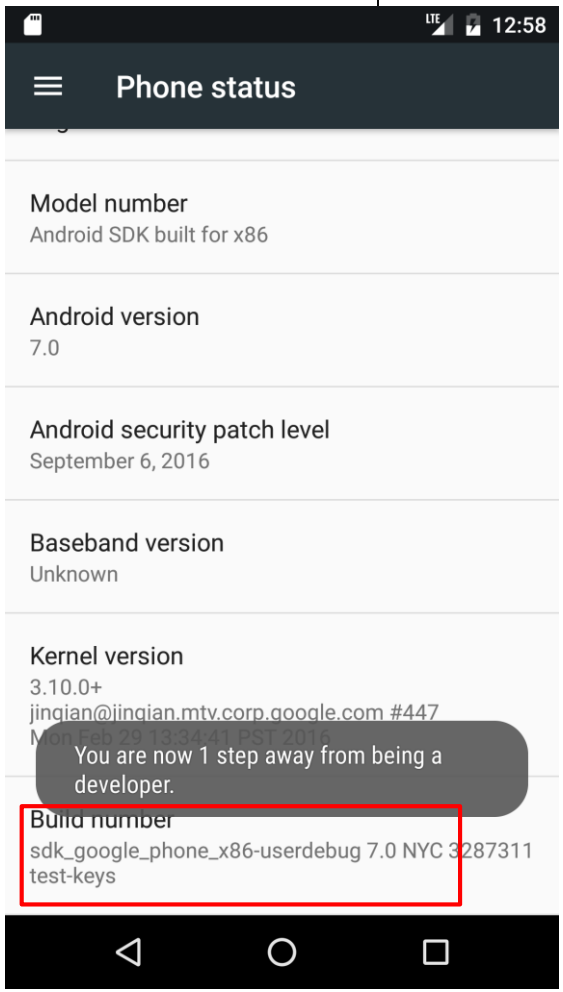




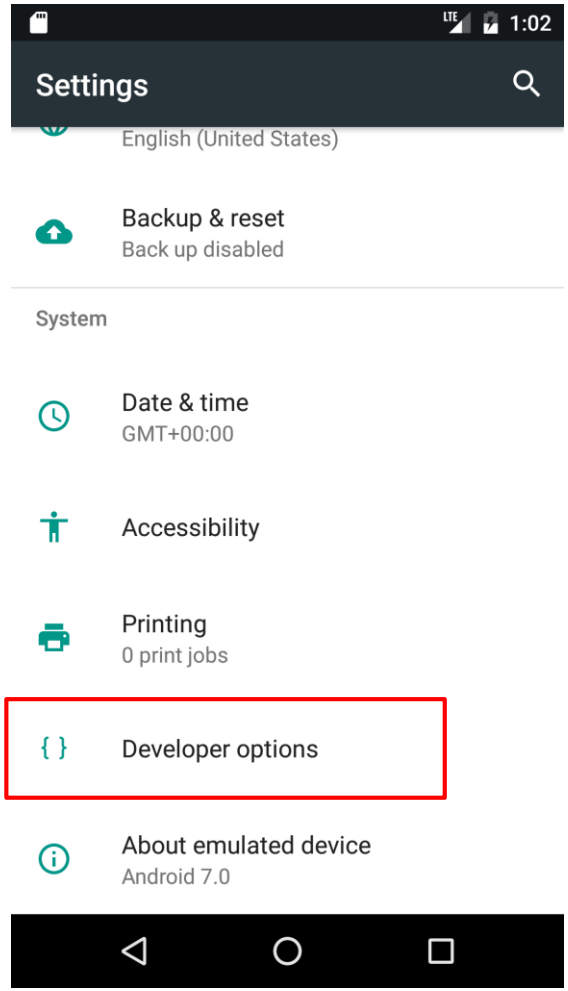
Configurer votre téléphone Android

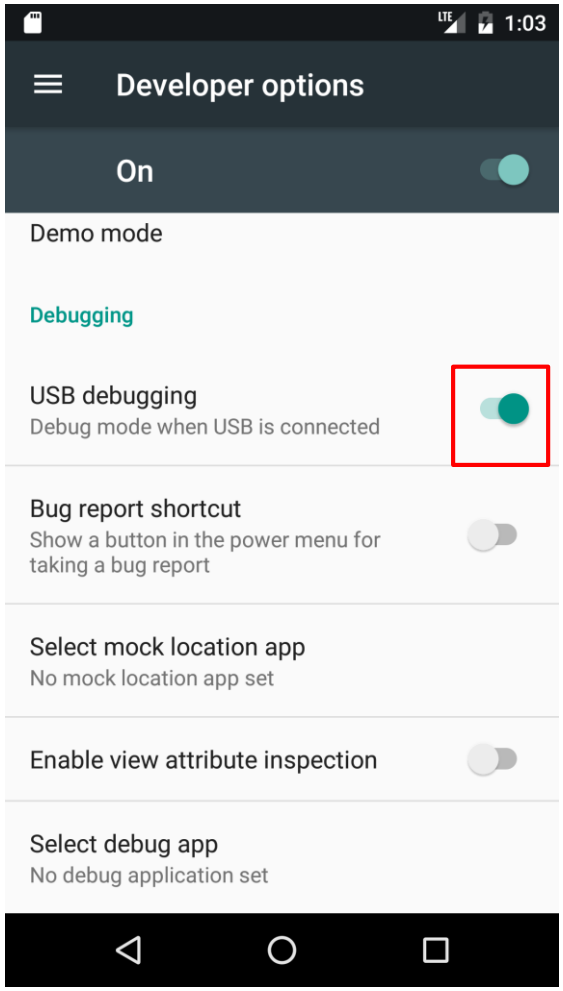




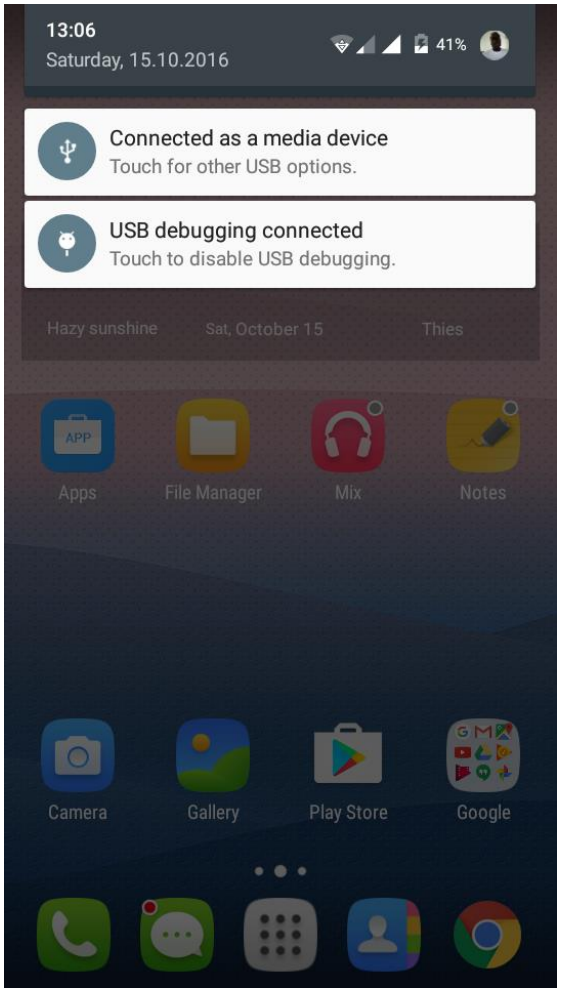


Configurer votre téléphone Android

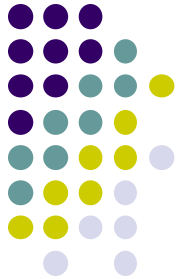




Mon téléphone après configuration



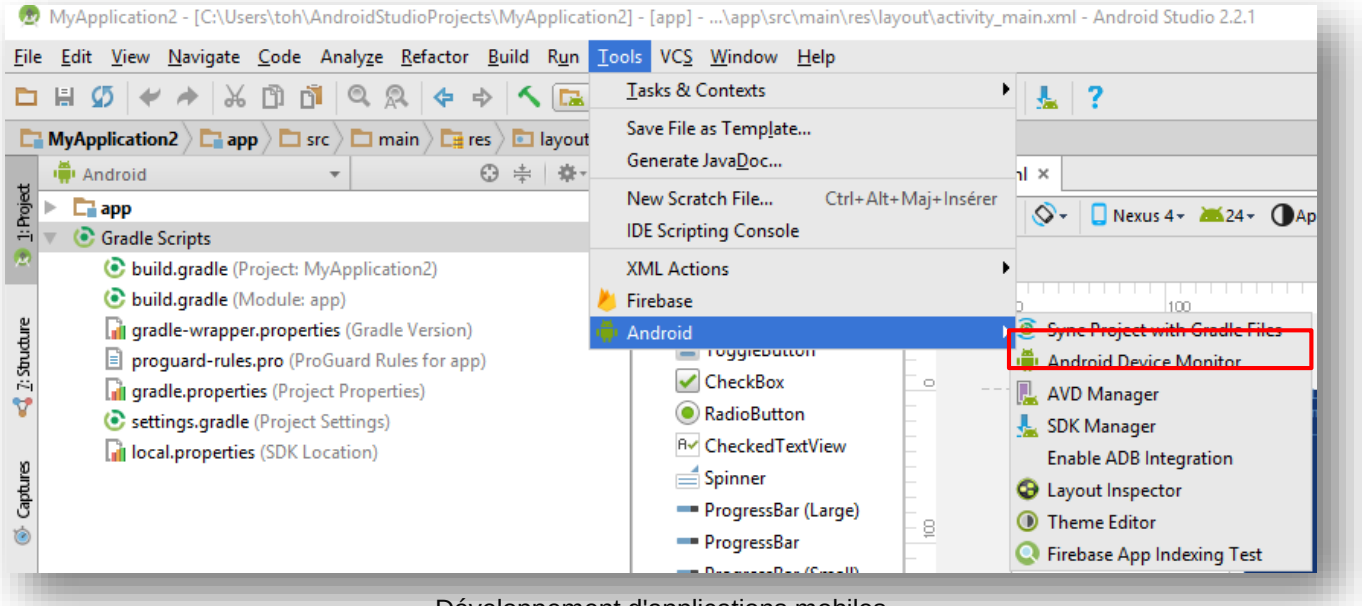
Evaluation



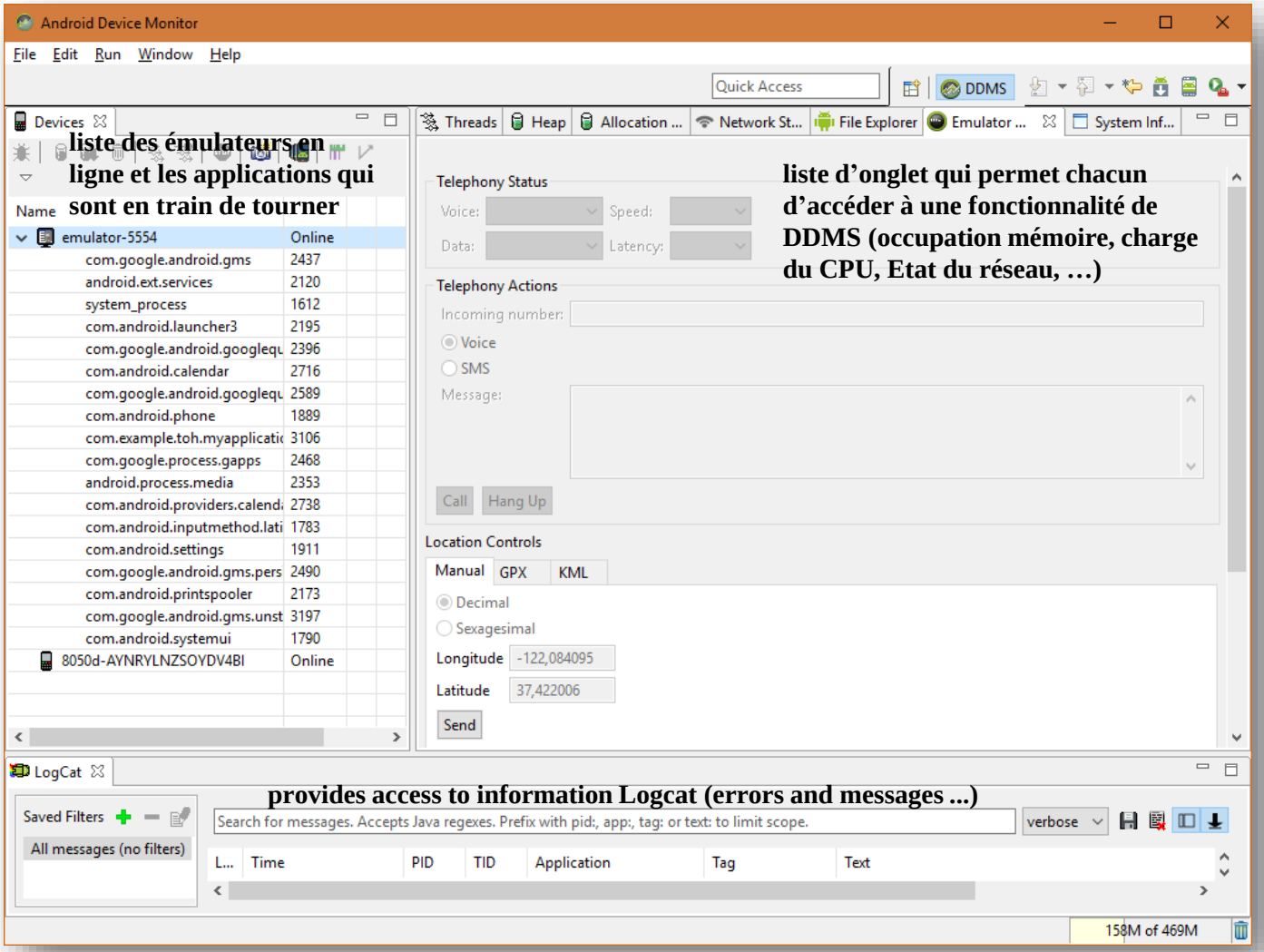
- Questions
  - Quels sont les différentes étapes pour créer un projet Android sous Android Studio
  - Rappelez le role des différents blocs d'Android Studio
  - Quels sont les prinpaux types de fichiers qu'on peut trouver dans l'application Android ?

Exercice : DDMS Exploration

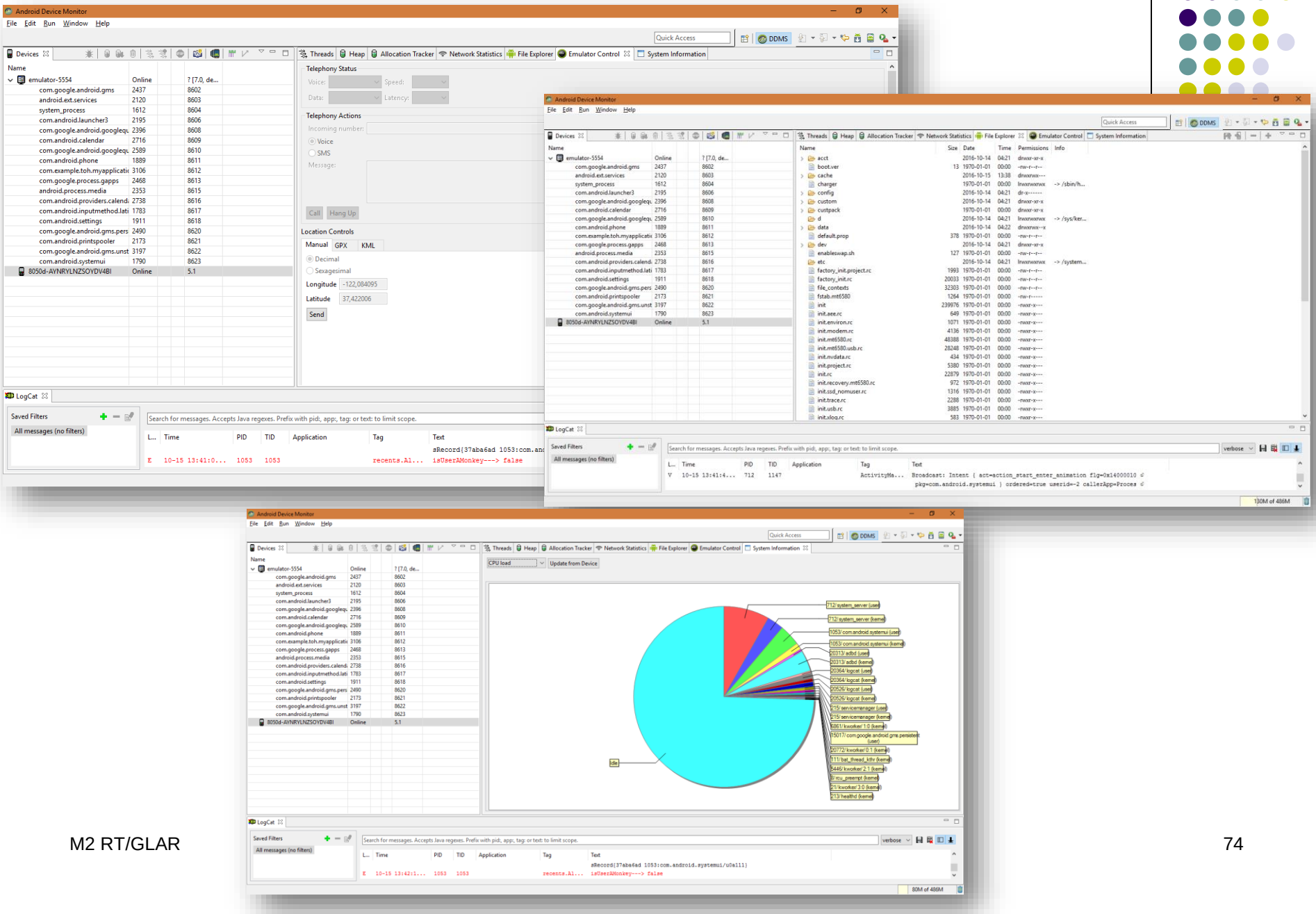
- **DDMS (Dalvid Debugger Monitor Service) ou AVD(Android Device Monitor)** est un outil qui permet de naviguer dans le système de fichier de l'émulateur, de voir les informations du système de fichier de l'émulateur, de simuler certaines fonctionnalités sur les émulateurs.
- Pour lancer le DDMS:

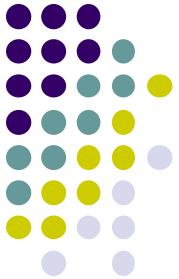


Exercice : Exploration du DDMS



Exercice : Exploration du DDMS





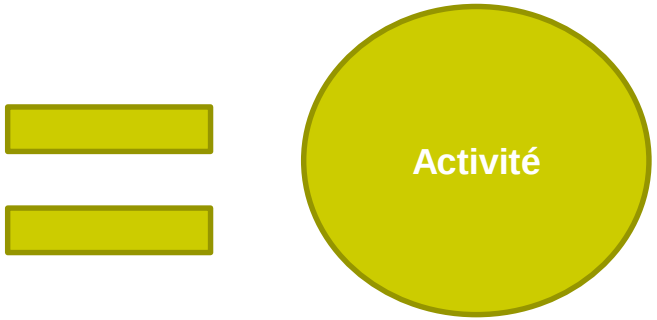
Quelques concepts autour des applications Android

Android application :

- Activité = Programme (Context) + Interface graphique

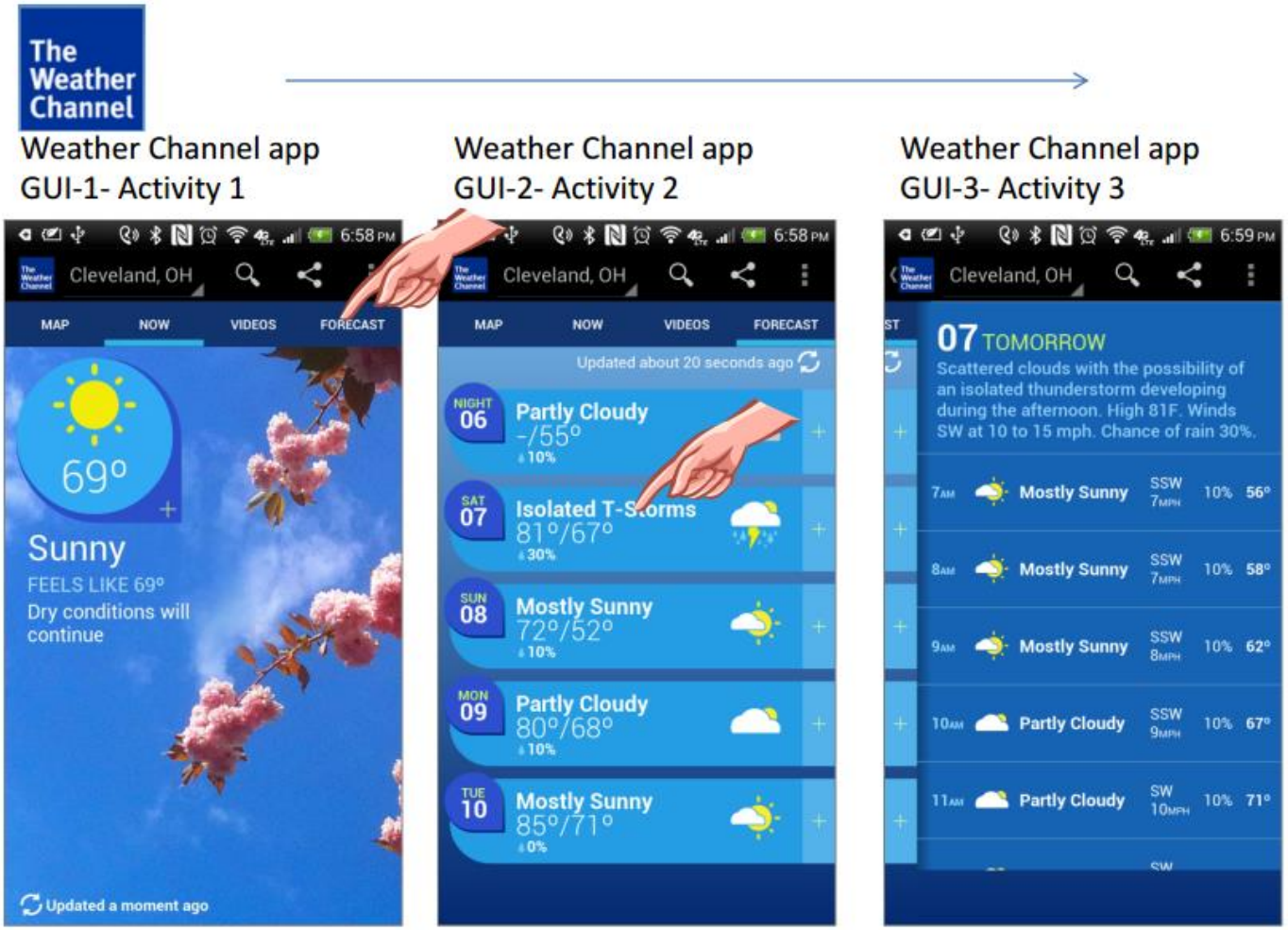
Context :
Informations sur l'état actuel de l'application (évènement, variable, méthode … etc)

Interface(s) Graphique(s)

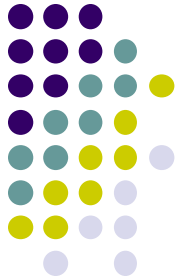


• A typical Android application consists of one or more activities.
• An activity is roughly equivalent to a Windows-Form .
• An activity usually shows a single visual user interface (GUI).
• Only one activity (known as main) is chosen to be executed first when the application is launched.
• An activity may transfer control and data to another activity through an interprocess communication protocol called intents.

Anatomie d'un application Android

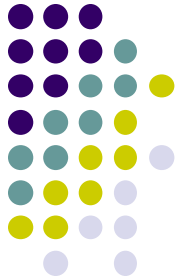


Services



- Un service est un composant qui fonctionne en tache de fond pour effectuer des opérations sur une longue durée ou effectuer un travail pour les processus distants.
- Un service ne fournit pas une interface utilisateur. Par exemple, un service peut jouer de la musique en arrière-plan pendant que l'utilisateur est sur une autre application, ou peut récupérer des données sur le réseau sans bloquer l'interaction de l'utilisateur avec une activité.
- Un autre composant, comme une activité, peut démarrer un service et le laisser tourner ou bien s’enregistrer à son niveau afin d’interagir avec elle. Un service est implémenté comme une sous-classe de Service

Intent



- Un ***Intent*** est un mécanisme pour décrire une action spécifique (imaginer un ***Intent*** comme un messager qui demande une action d'autres composants). Par exemple, il est possible de demander le lancement de l'une des applications capables de lire un fichier MP4 sans les nommer explicitement.
- L'intention est créé avec un objet **Intent** et peut être explicite ou implicite, respectivement.

AndroidManifest.xml file

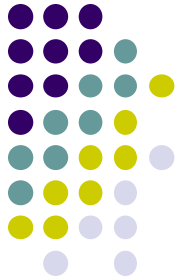
Généré par Eclipse, ce fichier contient la description de l'application

MainActivity.java × activity_main.xml × AndroidManifest.xml ×

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.toh.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

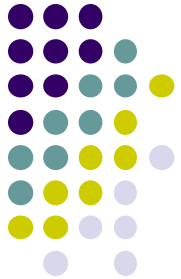
Il présente des informations sur l'application destinées au système Android.

Rubriques du fichier AndroidManifest



- Nom du package Java de l'application. Le nom du package est utilisé comme identifiant unique pour l'application.
- Description des composants de l'application - Activités (activités), services (services) récepteurs de radiodiffusion (de diffusion) et les fournisseurs de contenu (fournisseurs de contenu);
- Détermine le processus qui va accueillir les composants
- Déclarer les autorisations que l'application doit avoir afin d'accéder à des parties protégées de l'API et d'interagir avec d'autres applications.
- Identifier le niveau minimum de l'API Android requis par l'application.

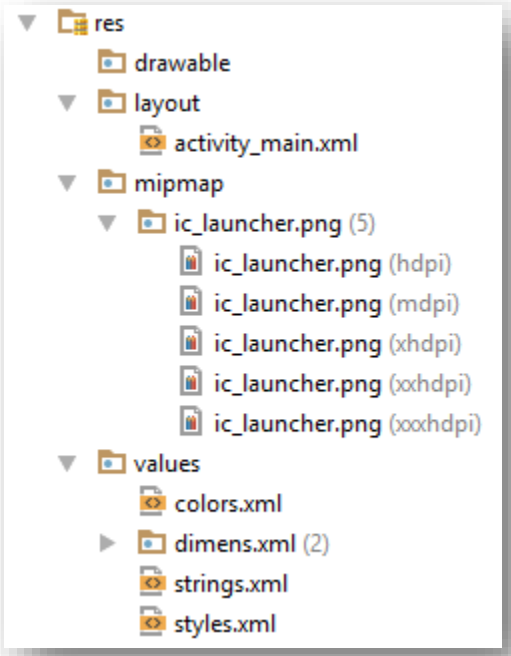
Rubriques du fichier AndroidManifest



- Ce fichier contient la description de l'application comprenant:
  - L'icône (**android: icon**)
  - Le nom nom de l'application (**android: label**)
  - Liste des activités qui compose l' application (**<activity>**)
  - Description des composants de l'application - Activités (activités), services (services) récepteurs de radiodiffusion (de diffusion) et les fournisseurs de contenu (fournisseurs de contenu), permissions…

Resources d'un projet Android

- Les ressources jouent un rôle important dans un projet Android. Une ressource est un **fichier** (texte, image, son, ou autre) ou une **valeur** qui est associé à une application exécutable.
- Ces ressources peuvent être modifiés sans recompiler l'application et ressemble beaucoup au concept de fichiers de configuration des applications classiques.
- Les ressources les plus courantes incluent : des chaines de caractères (texte sur les boutons), des couleurs (thèmes de couleurs), les images (icone de l'application, images sur l'application, …), …

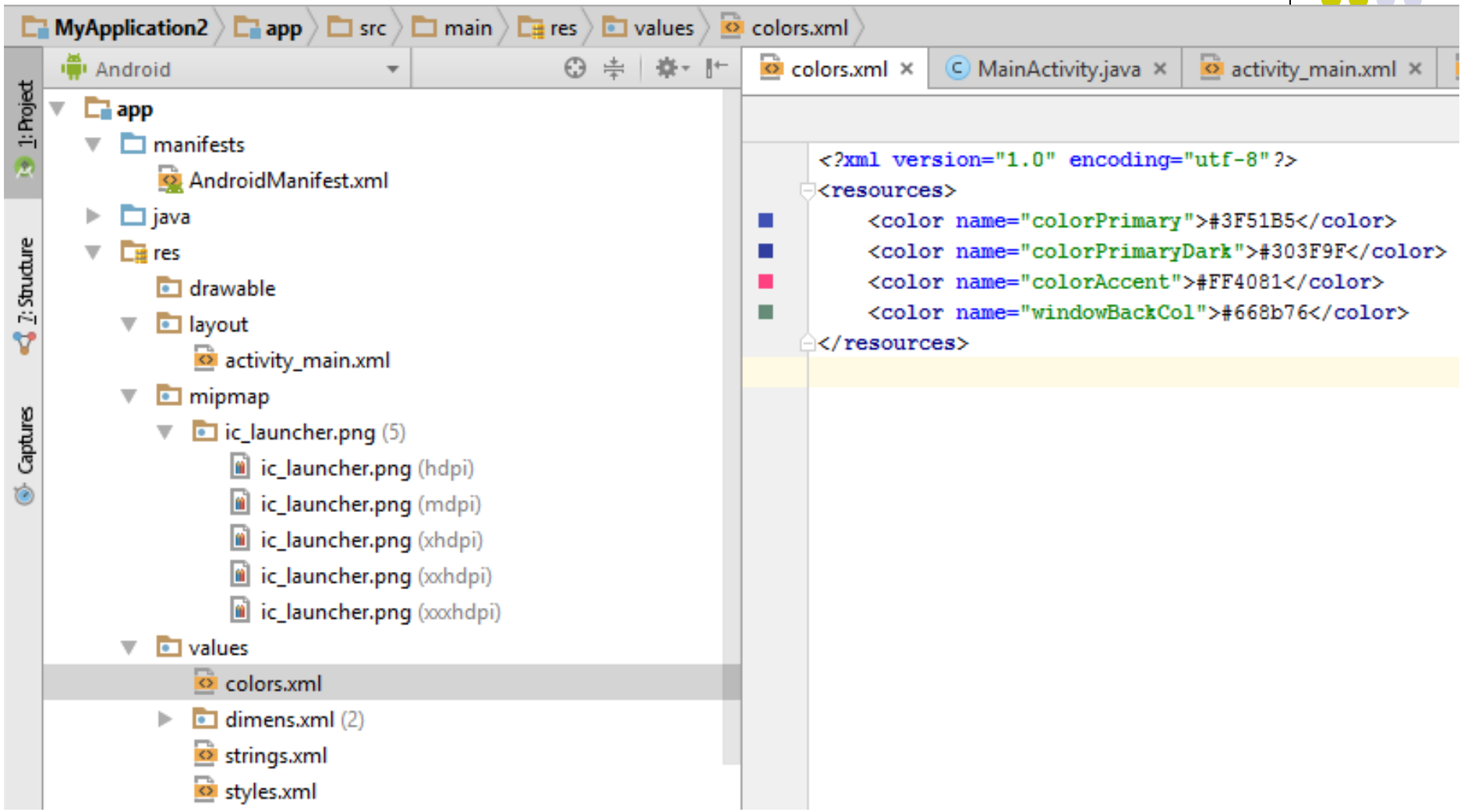


Les ressources sont stockées dans le répertoire **res/** du projet. Elles sont organisées dans des sous répertoires en fonction de leur rôle:

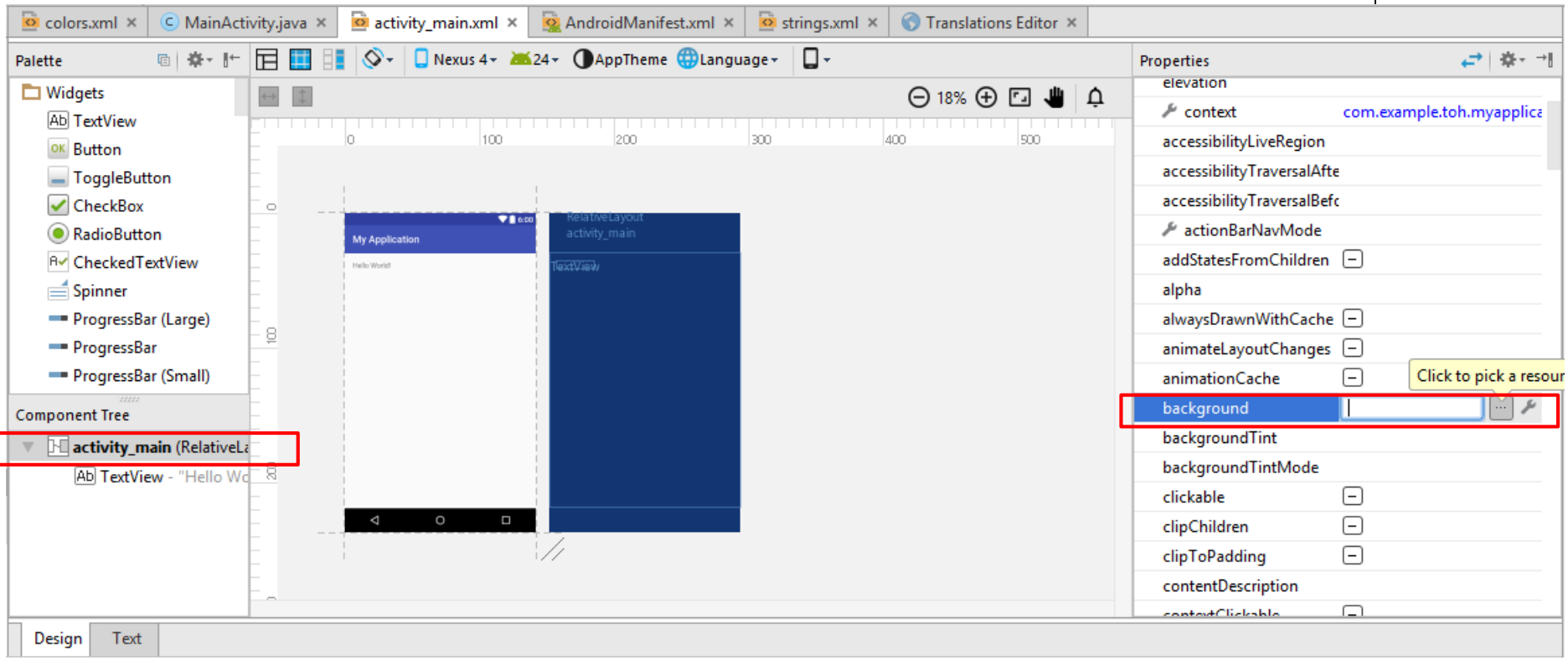
- **res/drawable** : Ce répertoire contient les images bitmaps ou XML (SVG).
- **res/layout** : Contient la définition de la présentation de l'interface de l'application (voir l'activité 2)
- **res/menu** : Décrit le contenu du menu
- **res/mipmap** : répertoire qui doit contenir l'icône de l'application avec différentes résolutions.
- **res/values** : définit des chaînes de caractères, des tableaux de chaînes de caractères (y compris le formatage de chaînes et le style

Demonstration avec string.xml and color.xml

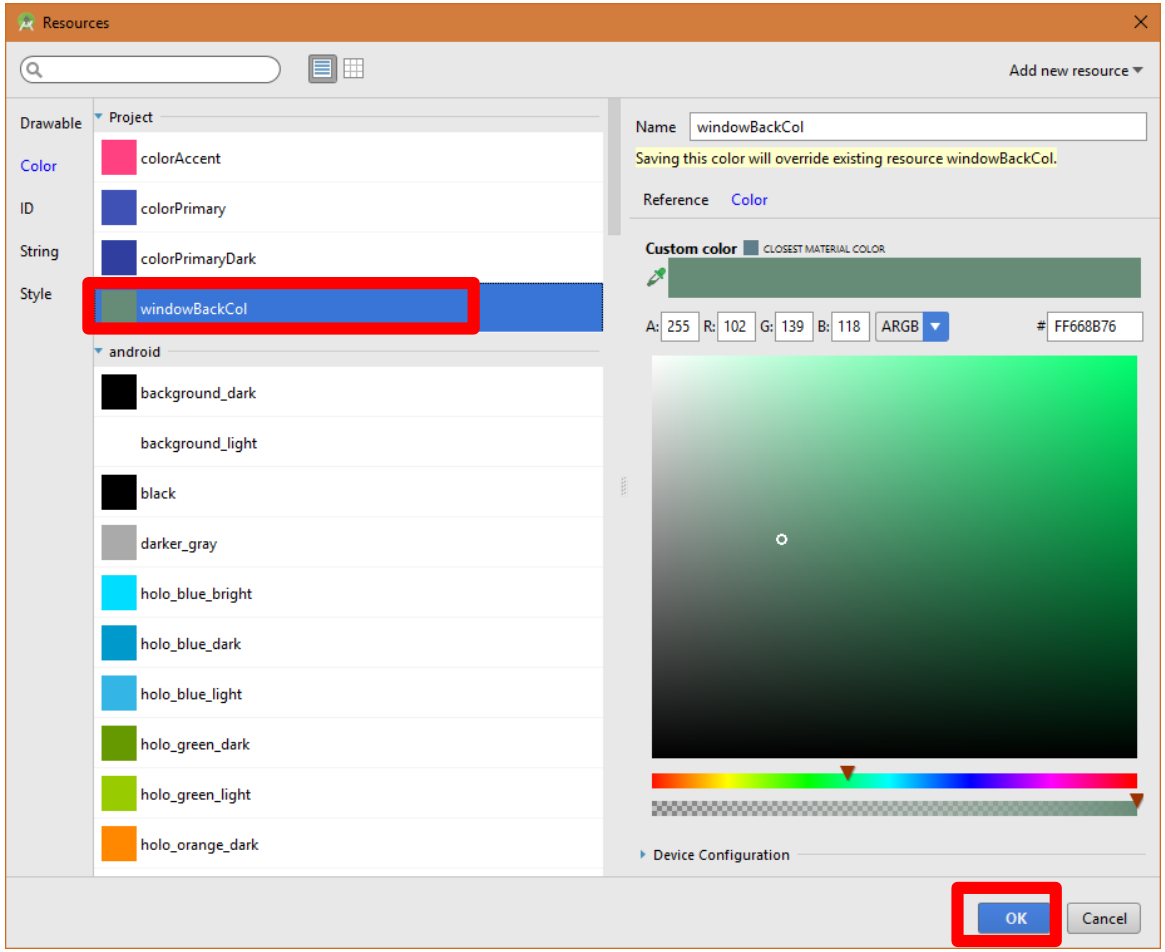
Ressource Color



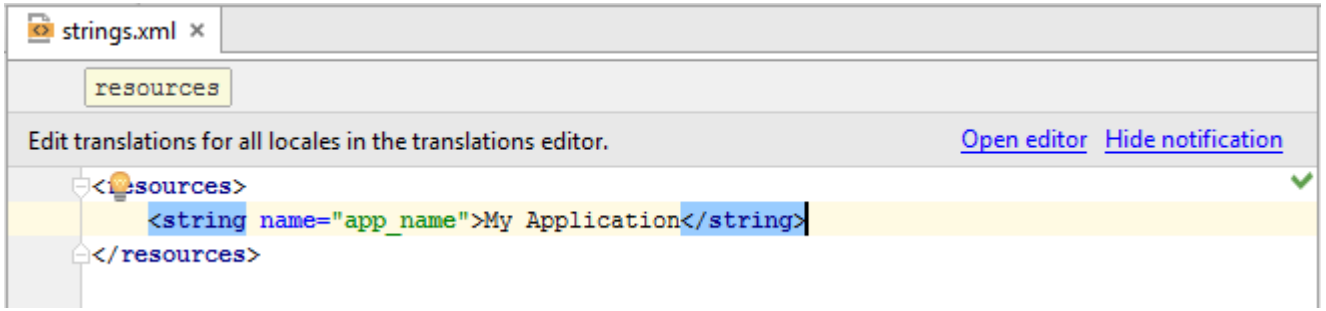
Changement de la couleur de fond de la fenêtre

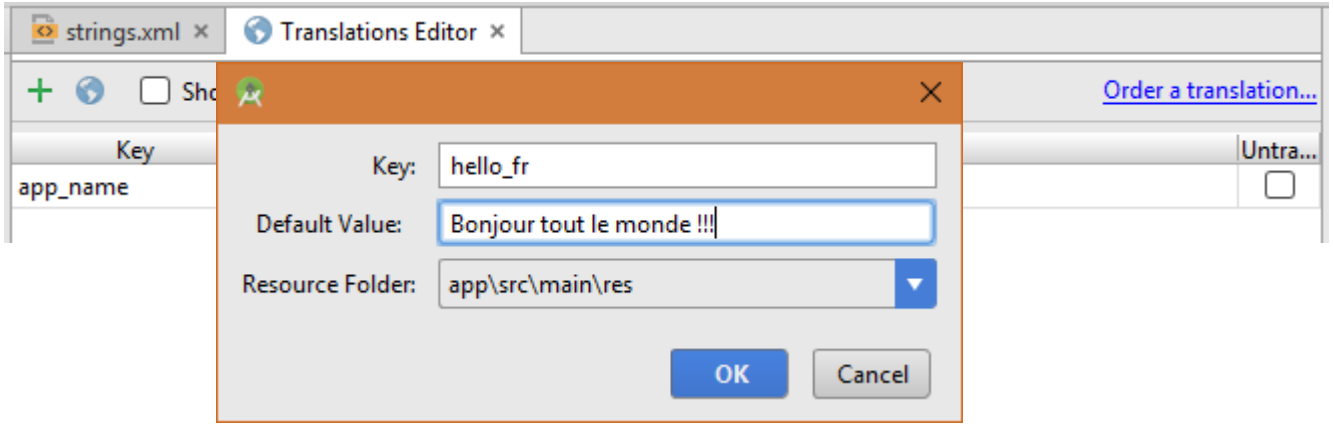


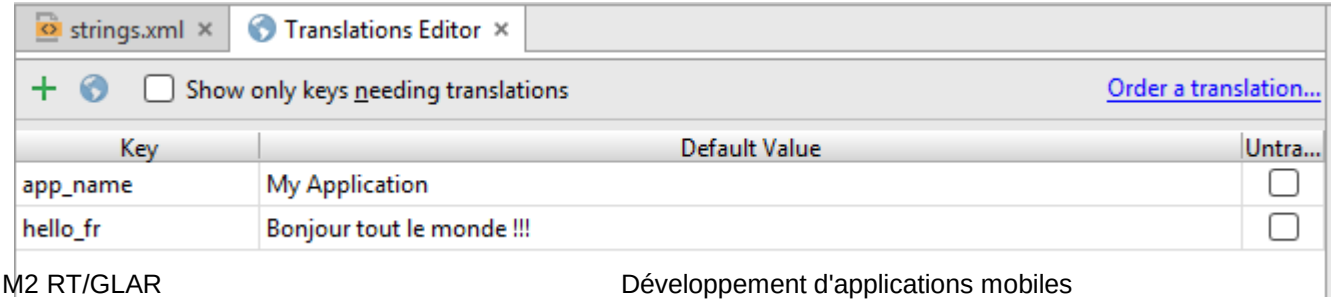
Changement de la couleur de fond de la fenêtre



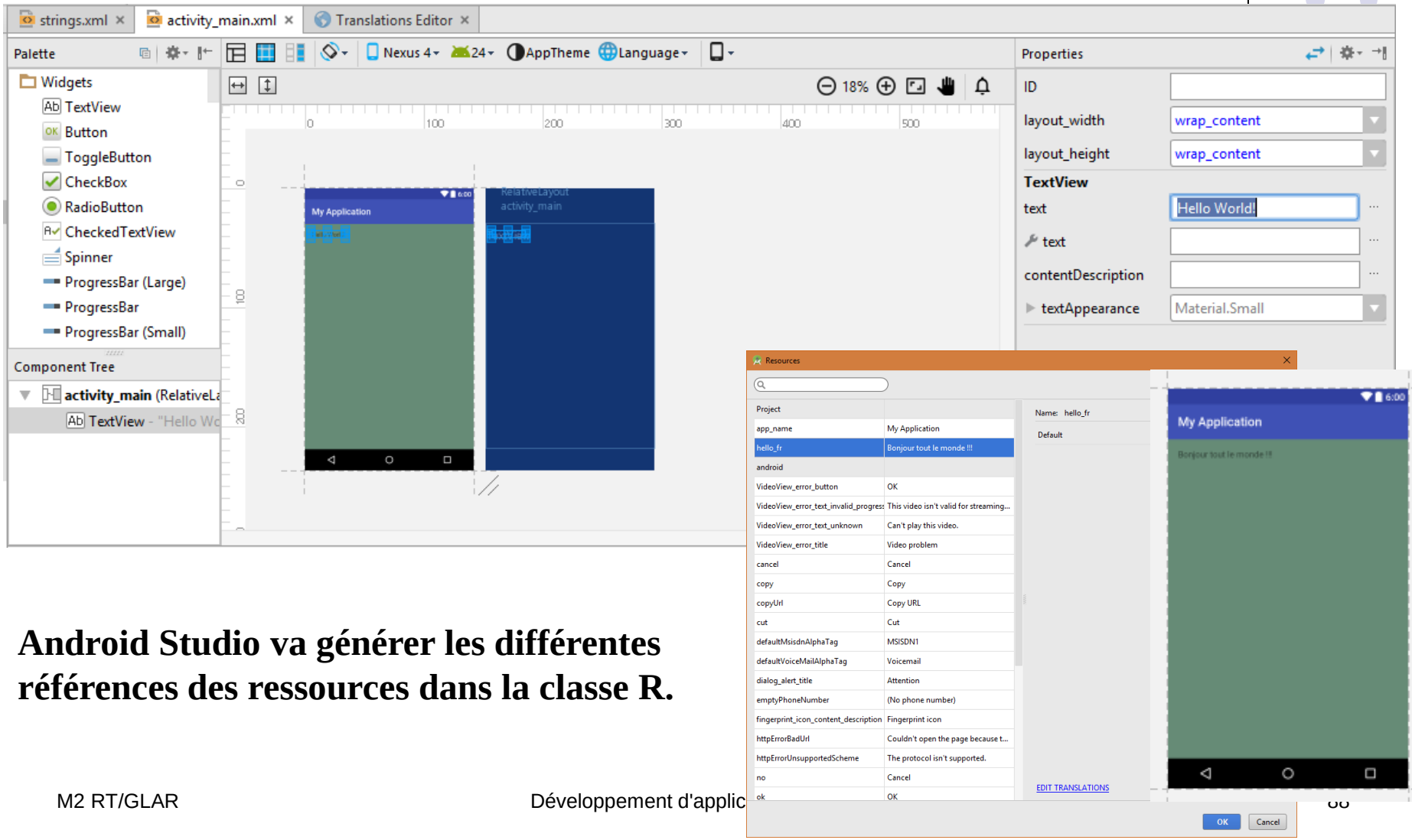
Creation d'une ressource de type string dans string.xml







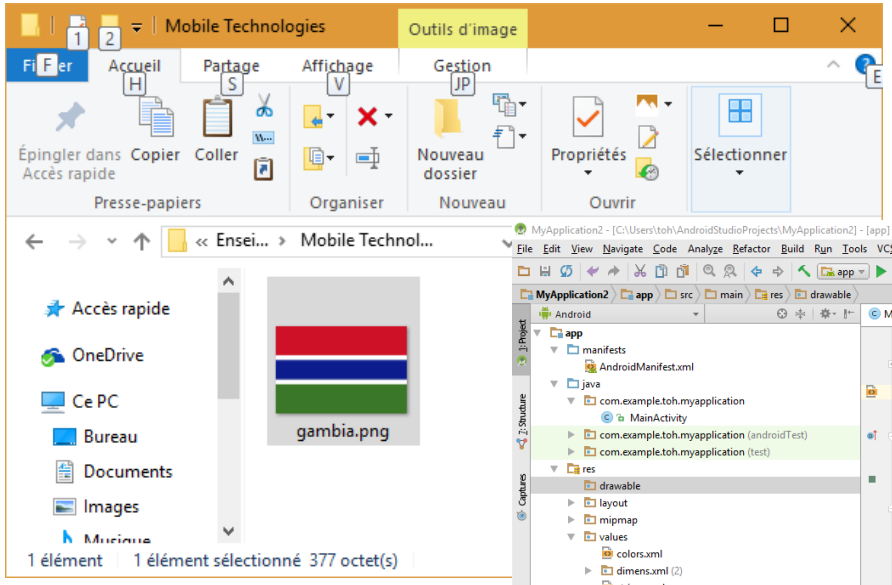
Creation d'une ressource de type string dans string.xml



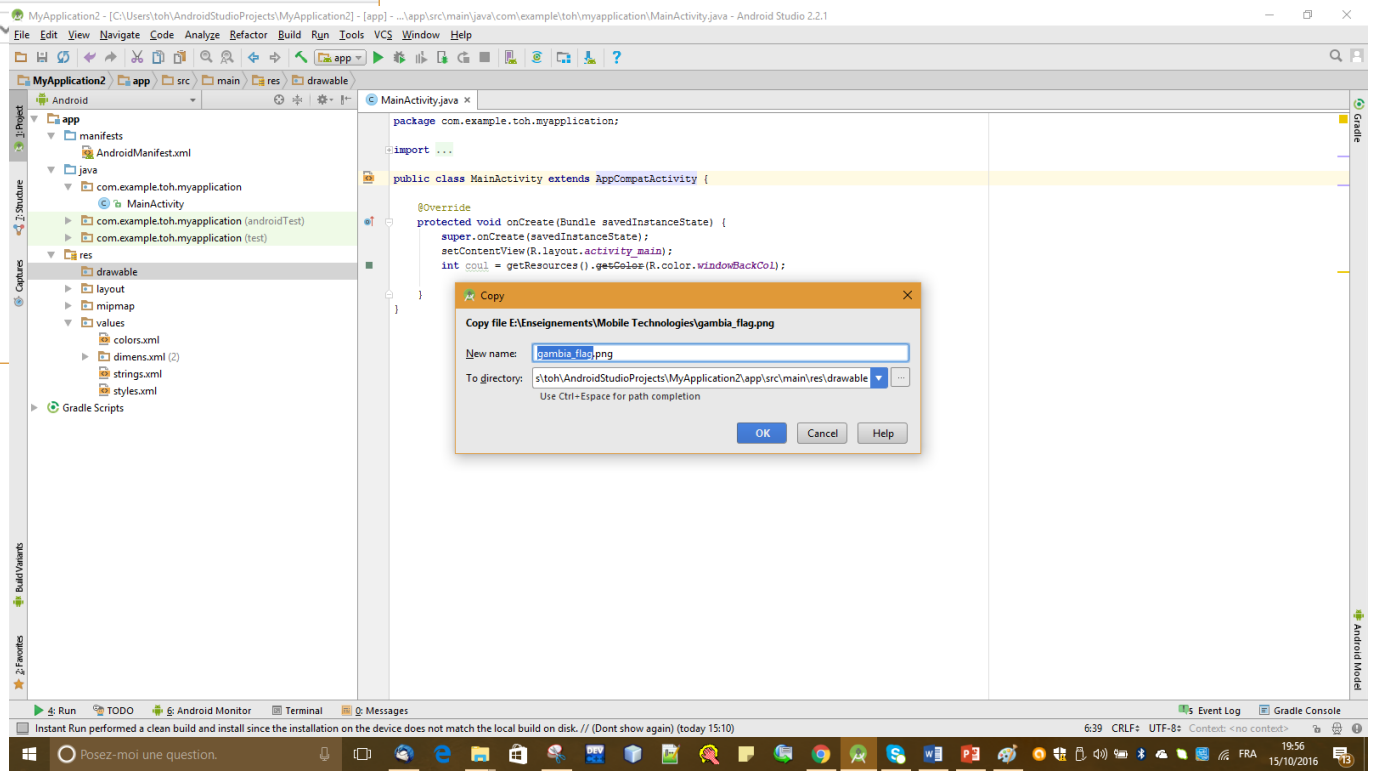
Android Studio va générer les différentes références des ressources dans la classe R.

Ressource Drawable

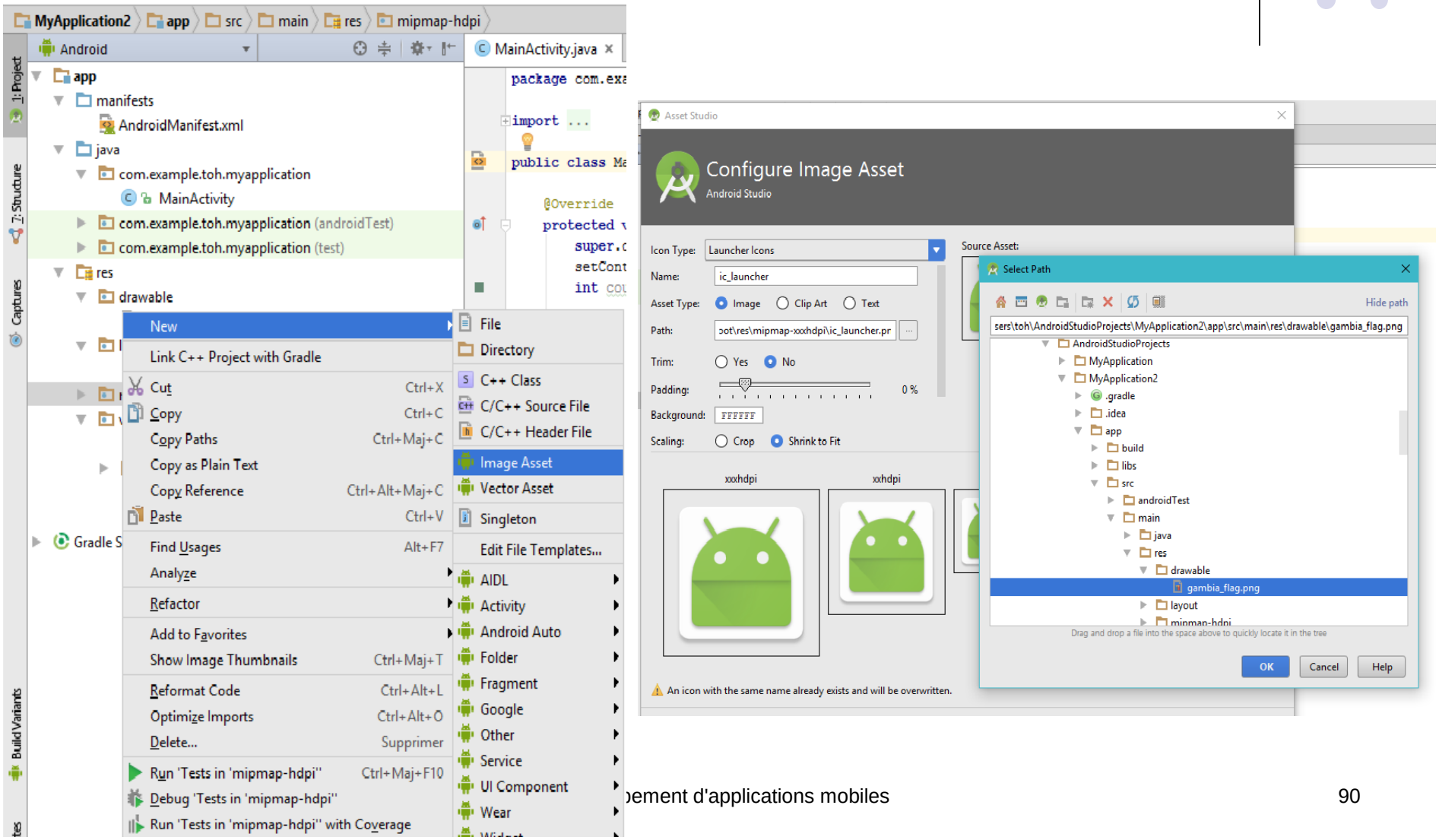
- Copier une image de votre ordinateur



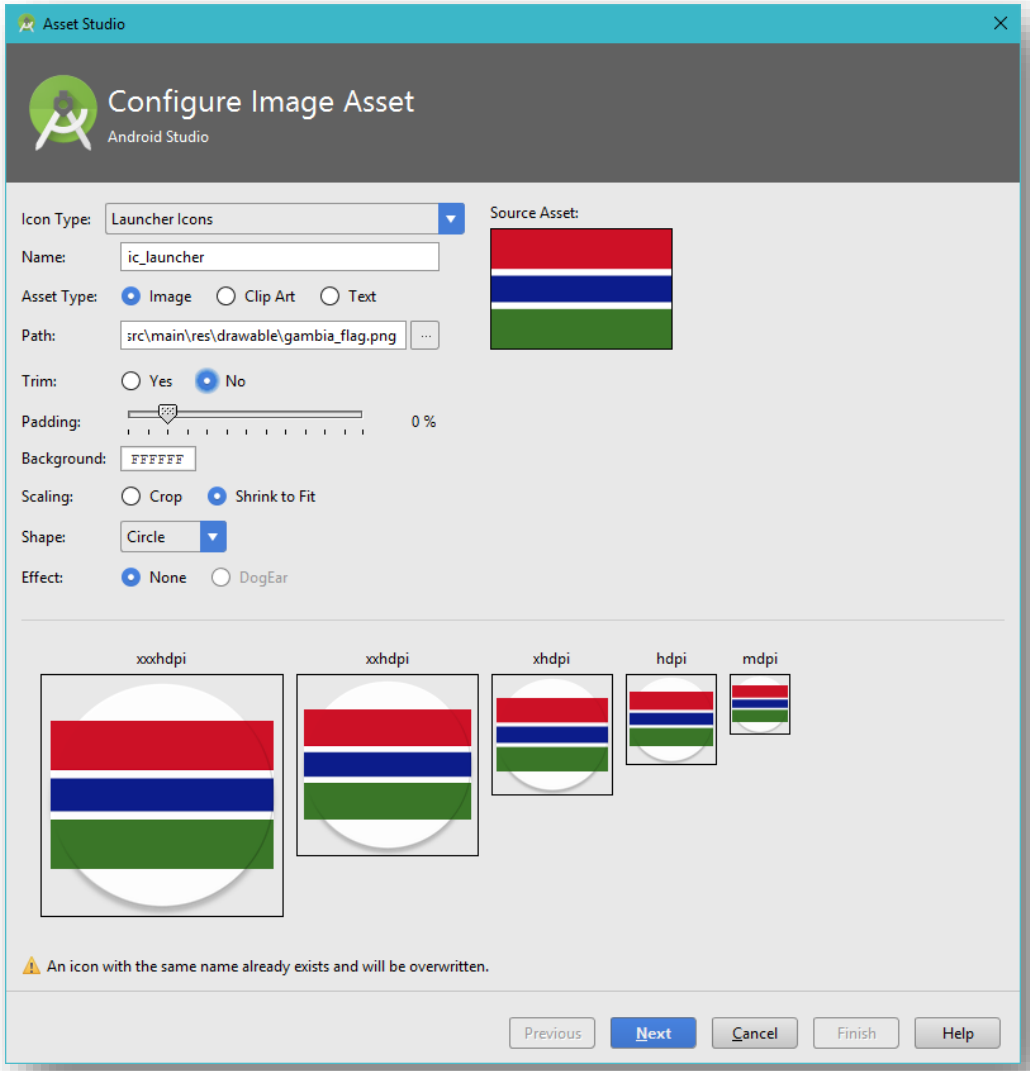
Coller dans le répertoire drawable



Ressource Drawable : Changer l'icone de l'application

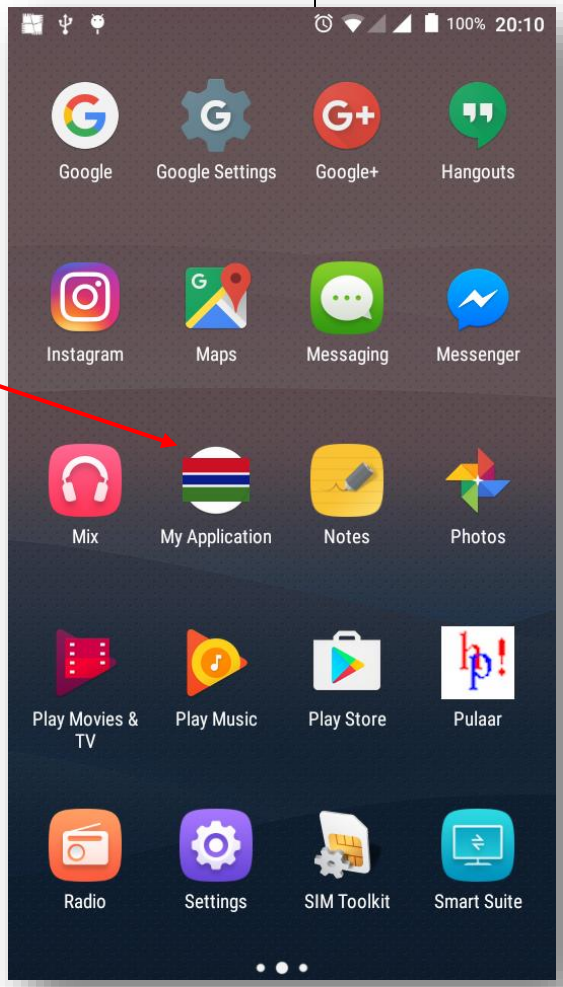


Ressource Drawable : Changer l'icone de l'application



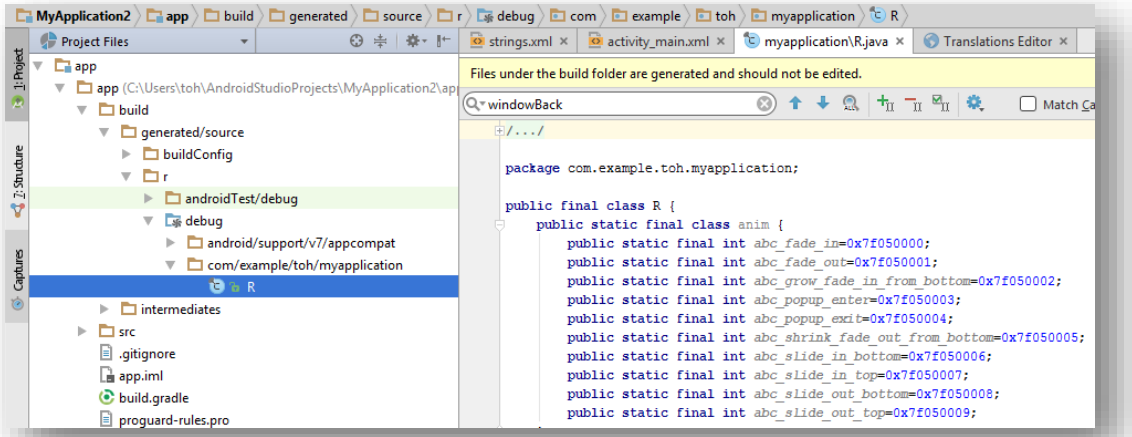
Ressource Drawable : Changer l'icone de l'application

- Si vous charger l'application vous pourrez contater que son icône a changé.

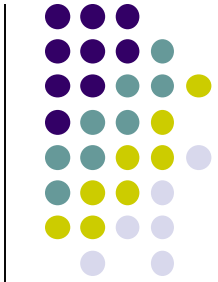


La classe R

- C'est une classe générée par Eclipse qui permet à l'application d'accéder aux ressources
- Elle contient des classes internes dont les noms correspondent aux types de ressources (id, drawable, layout …)
- Elle est constituée à partir des fichiers placés dans les sous répertoires du répertoire res
- Une propriété est créée pour :
  - Chaque image placé dans drawable-xxxx
  - Chaque identificateur défini dans des fichiers XML (objets d'interface, constantes)
  - Chaque fichier placé dans les répertoires xml , raw …



Utilisation des ressources



- Référencement d'une ressource dans un fichier xml. La forme générale est : "@type/identificateur"

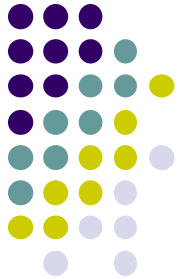
  Par exemple : @string/*machaine* fait référence à une chaine contenue dans un fichier XML placé dans le répertoire **res/values** et définie comme suit :

```
<resources>
    …
    <string name="machaine">Contenu de cette chaine</string>
    …
</resources>
```

- Référencement d'une ressource dans le code. La forme générale est : R.type.identificateur

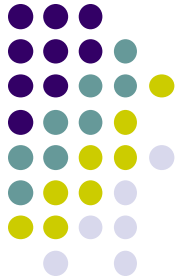
  Par exemple : R.string.*machaine* fait référence à la même chaine

La classe Resources



- Permet l'accès aux ressources répertoriées dans R
- On obtient une instance de cette classe par getResources() de l'activité
- Principales méthodes de la classe Resources (le paramètre est un identifiant défini dans R de la forme R.type.nom) :
  - boolean getBoolean(int)
  - int getInteger(int)
  - int[] getArray(int)
  - String getString(int)
  - String[] getStringArray(int)
  - int getColor(int)
  - float getDimension(int)
  - Drawable getDrawable(int)
- Exemple : String titre = getResources().getString(R.string.ma_chaine);
  String coul = getResources().getColor(R.color.ma_chaine);

Utilisation des ressources



- Accès aux ressources dans l'application
  - Mise en place de l'interface principale
    - setContentView(R.layout.nom_du_fichier_xml);
  - Mise en place d'interfaces supplémentaires
    - Par les classes LayoutInflater ou MenuInflater
  - Accès direct à une valeur ou à une ressource :
    - String titre = getResources().getString(R.string.texte_titre);
    - Drawable monImage = getResources().getDrawable(R.drawable.nom_de_l_image)

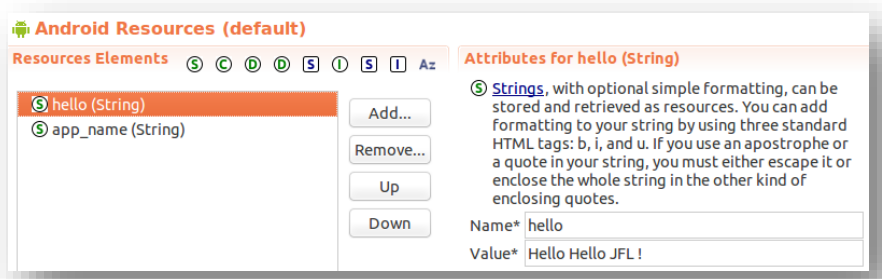
Les chaines

- Les chaines constantes de l'application sont situées dans **res/values/strings.xml.** Voici un exemple:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello Hello CFPP !</string>
    <string name="app_name">AndroJF</string>
</resources>
```

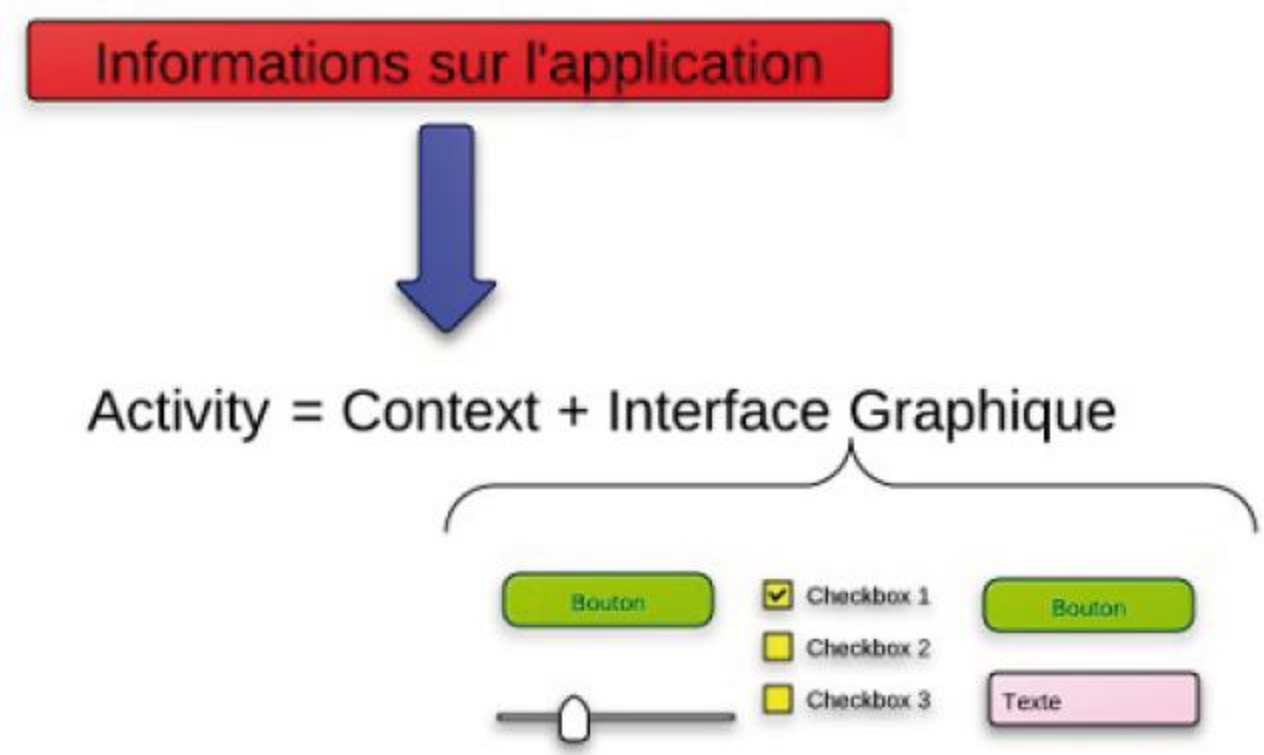
- La récupération de la chaine se fait via le code:

```
Resources res = getResources();
String hw = res.getString(R.string.hello);
```



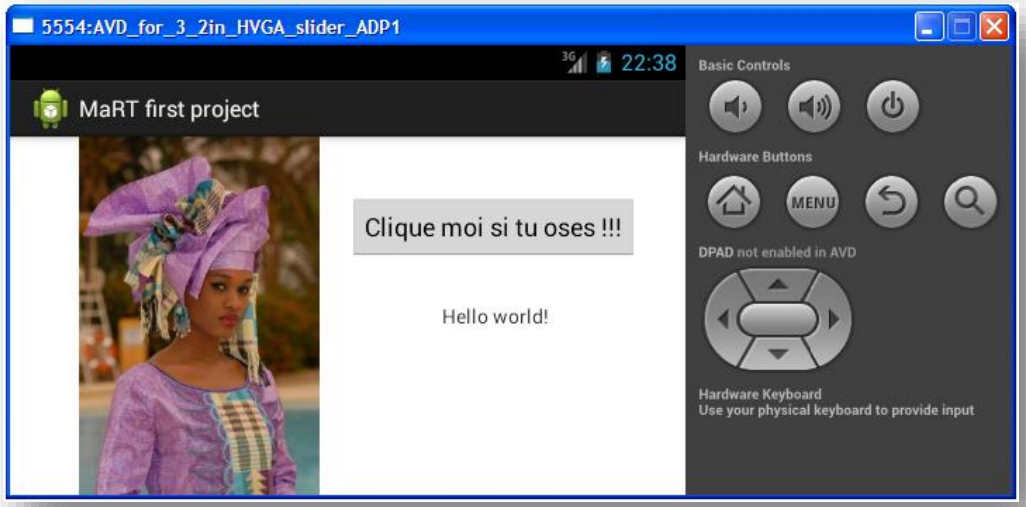
Application Android

- Une activité = une interface + un programme



Activité Android

- Partie importante de l'interface graphique Android
- Peut être considéré comme une «fenêtre»



Activité Android

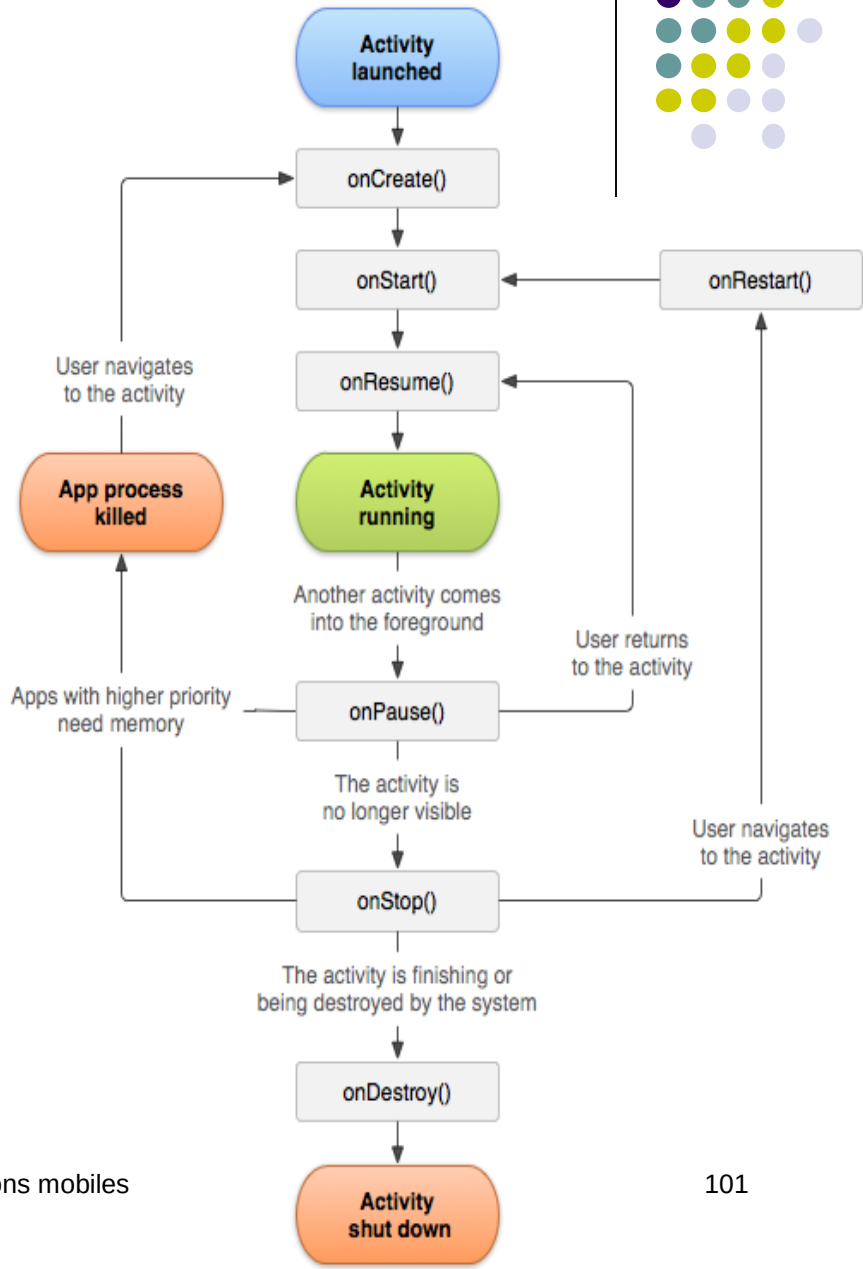
- Une activité est le composant d'application qui fournit une fenêtre (un écran) à travers laquelle les utilisateurs peuvent interagir et initier une action, telles que la composition d'un numéro de téléphone, envoyer un e-mail, visionner une vidéo, etc.

- Les activités sont un élément fondamental dans le développement d'applications Android. Ils peuvent exister dans un certain nombre d'états différents. Le cycle de vie de l'activité commence par son instanciation et se termine par sa destruction, et passe par de nombreux états intermédiaires.

- Quand une activité change d'état, la méthode de l'événement du cycle de vie approprié est appelée afin d'avertir l'activité du changement imminent de son état. Ces callbacks lui permettent d'exécuter du code afin de l'adapter à ce changement.

Cycle de vie d'une activité

Callback	Description
onCreate()	Cette fonction est la première qui est exécutée lorsque l'activité est créée pour la première fois.
onStart()	Ce callback est appelé lorsque l'activité devient visible à l'utilisateur.
onResume()	Cette méthode est appelée lorsque l'utilisateur commence à interagir avec l'application.
onPause()	Une activité en pause ne reçoit pas les saisies de l'utilisateur et ne peut pas exécuter de code. Ce callback est appelée lorsque l'activité en cours est en pause et l'activité précédente a repris.
onStop()	Ce callback est appelé lorsque l'activité n'est plus visible.
onDestroy()	Ce callback est appelé avant que l'activité est détruite par le système.
onRestart()	Ce callback est appelée lorsque l'activité redémarre après un arrêt.



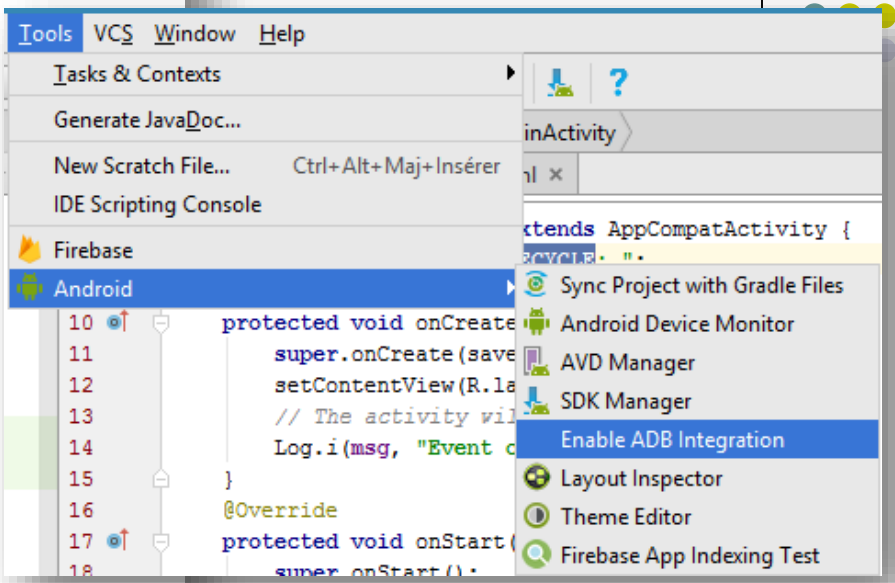
Cycle de vie d'une activité

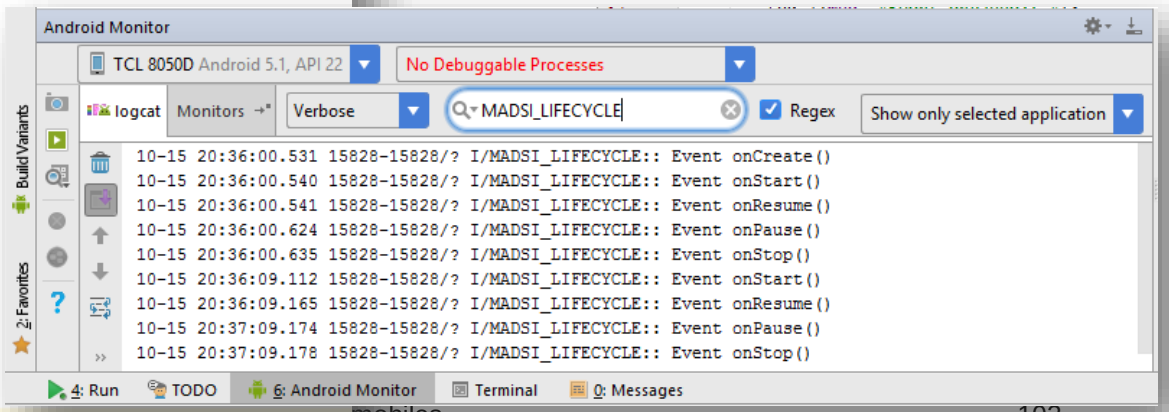
- Pour créer une activité, il suffit d'étendre la classe d'activité (ou une de ses sous classes et ré-implémenter les méthodes **onStop(), onStart(), onCreate(), onResume(), OnPause(), OnDestroy()**.
- Le code suivant donne le squelette typique pour une activité.

MainActivity.java × activity_main.xml ×

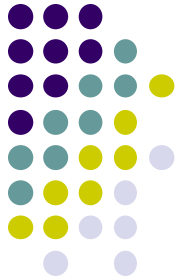
```
package com.example.toh.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

```java
public class MainActivity extends AppCompatActivity {
    String msg = "MADSI_LIFECYCLE: ";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // The activity will be created.
        Log.i(msg, "Event onCreate() ");
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity will be visible.
        Log.i(msg, "Event onStart() ");
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity will become visible again ...
        Log.i(msg, "Event onResume() ");
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another application will take focus.L'activité be paused
        Log.i(msg, "Event onPause() ");
    }
    @Override
    protected void onStop() {
        super.onStop();
        // The activity is not possible
        Log.i(msg, "Event onStop() ");
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity will be destroyed.
        Log.i(msg, "Event onStop()");
    }
}
```





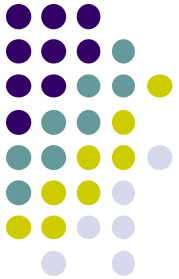
Conclusion



- Une application Android :
  - Série d’activités qui “vivent" et “meurent"
  - Une activité est généralement composée d’une interface graphique et d’un context (programme)

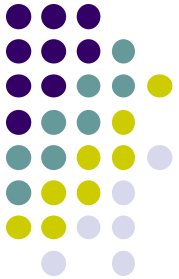
- Une application ne limite pas le nombre d’activités mais le périphérique(émulateur par ex.) si.

Evaluation



- Quels sont les différents types de fichiers dans un projet Android ?
- Remplissez le tableau suivant :

Callback	Evènement déclencheur de l'appel
onCreate()	
onStart()	
onResume()	
onStop()	
onDelete()	



LET’S GO !!

Développement d'applications mobiles

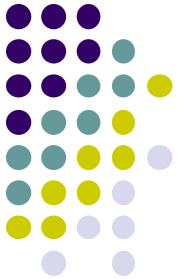
Android - Création d'interfaces graphiques

M2 RT/GLAR

PARTIE 3

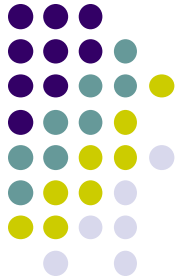
Pr. El hadji M. NGUER

Pensez vos interfaces pour un smartphone



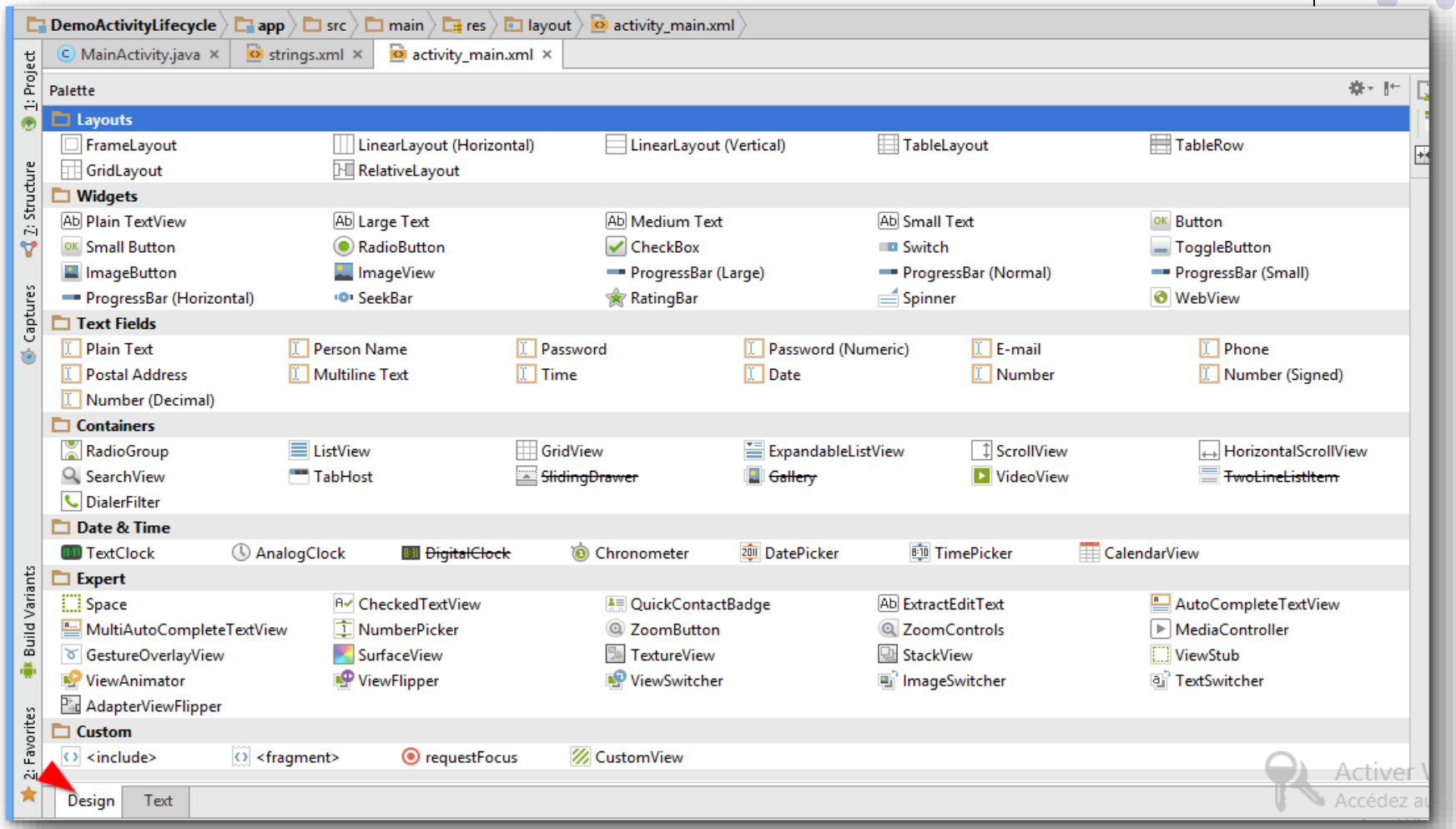
- Ecran tactile de petite taille
  - Eviter les interfaces trop touffues (on ne peut pas agrandir l'écran comme on agrandit une fenêtre)
  - Eviter les éléments cliquables trop petits (il faut pouvoir cliquer avec le doigt même si on a des gros doigts)
  - Eviter les élément cliquables trop tassés (il faut pouvoir cliquer sur le bon élément même si on vise mal)
- Le défilement se fait par touché/glissé
  - Pas trop d'ascenseurs (on ne peut pas faire défiler un conteneur entier ET des éléments de ce conteneur dans le même sens)
  - Pas d'ascenseurs mal placés (si tous les éléments sont cliquables comment faire défiler sans cliquer ?)
- L'écran peut être tourné
- Tous les smartphones n'ont pas la même définition d'écran

Création d'interfaces

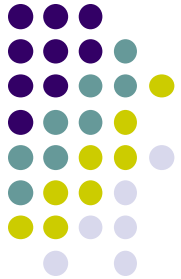


- Par programme (comparable à java swing) mais avec des classes propres à Android
  - Définition de conteneurs (un conteneur = un conteneur + un mode de placement = JPanel + Layout)
  - Définition d'éléments d'interaction (widgets) + placement et ajout dans les conteneurs
- Par description dans des fichiers xml (forme déclarative statique)
- Une interface est un arbre dont la racine est l'écran et les feuilles les éléments de l'interface (boutons, textes, cases à cocher, …)

Liste des vues Android sur l'interface de conception d'Android Studio

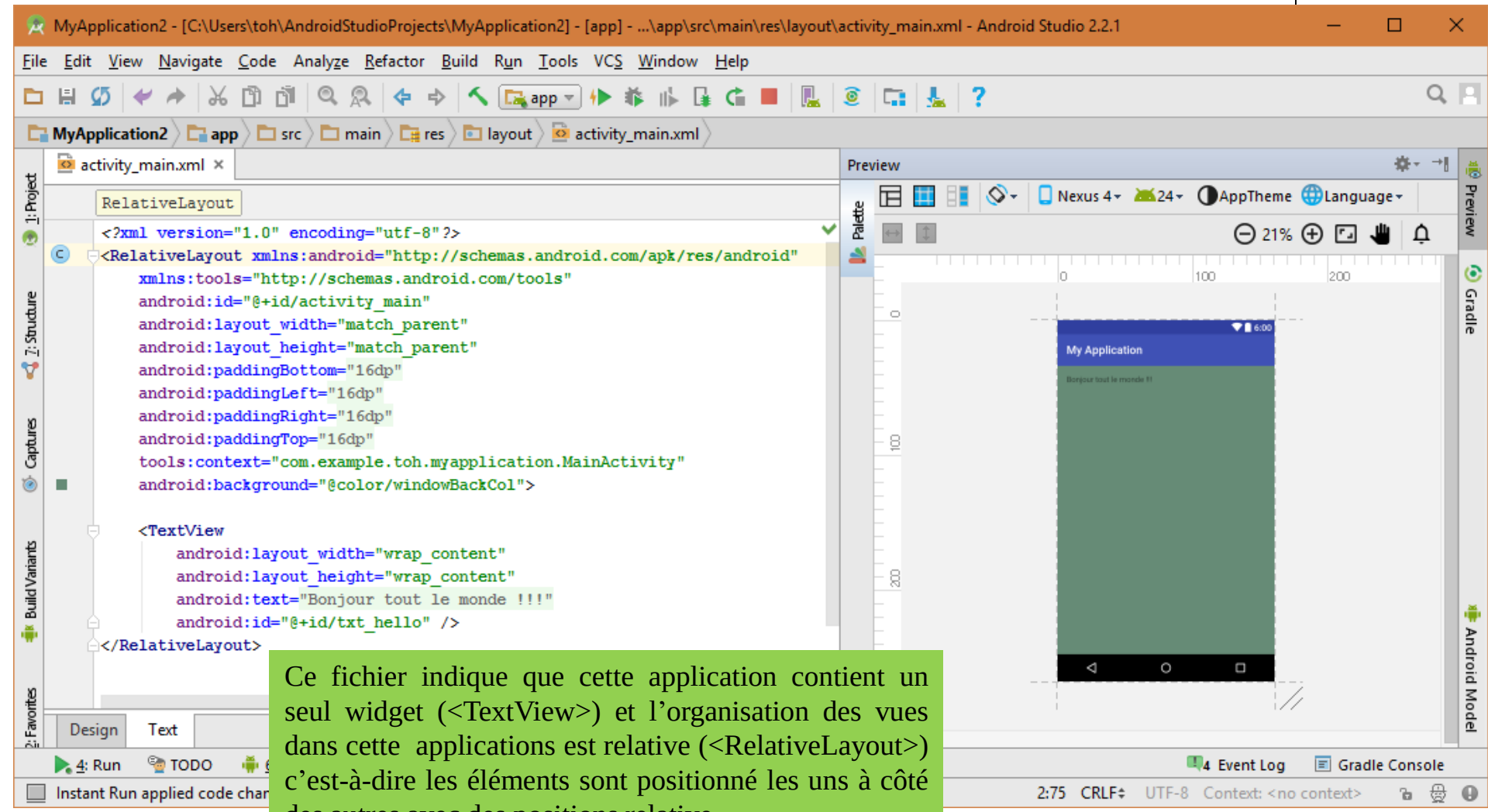


Codage de l'interface utilisateur



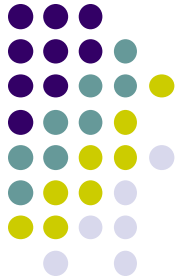
- Un layout définit la structure visuelle pour une interface utilisateur, tels que l'interface utilisateur pour une activité ou widget. Il est de déclarer une disposition de deux manières:
  - Une déclaration en XML. Android fournit un vocabulaire XML simple qui correspond à une instance de la classe View ou de ses sous-classes. Sous Android Studio, les layouts sont déclarés dans les fichiers dans **res\layout**. Le fichier par défaut est **res\layout\activity_main.xml**
  - Une instanciation des éléments du layout à l'exécution. Une application Android peut créer des objets de la classe View et ViewGroup (et manipuler leurs propriétés) par programmation. Sous Android Studio, cette option est gérée par dans les fichiers sources Java des activités (java/*) : Il s'agit ici d'ajouter des vues par programmation Java

Declaration XML: example de Activity_main.xml



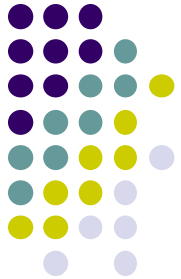
Ce fichier indique que cette application contient un seul widget (<TextView>) et l'organisation des vues dans cette applications est relative (<RelativeLayout>) c'est-à-dire les éléments sont positionné les uns à côté des autres avec des positions relative.

Unités de mesure dans les fichiers XML



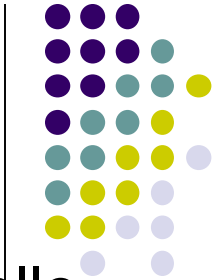
- Dans les fichiers XML, les dimensions des éléments d'interface (taille, marges, …) peuvent être exprimées en diverses unités :
  - Pixels (px)
  - Pouces (in)
  - Millimètres (mm)
  - Points (pt) = 1/72 pouce
  - Pixel à densité indépendante (dp) 1 dp = 1 pixel pour un écran de 160 dpi
  - Pixel à taille indépendante (sp) relatif à la taille des polices de caractères
- Dans les fichiers XML les unités sont exprimées sous la forme : "24.5mm" ou "65px" …

Couleurs dans les fichiers XML



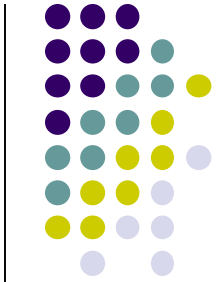
- Dans les fichiers XML, les couleurs sont exprimées sous la forme d'une chaîne de caractères codant les composantes en hexadécimal : "#AARRVVBB"
  - AA est l'opacité (00 totalement transparent, FF opaque)
  - RR est la composante rouge (00 à FF)
  - VV est la composante verte (00 à FF)
  - BB est la composante bleue (00 à FF)
- Si AA est omis la couleur est opaque

Android Layout Managers



- Android offre différents conteneurs de mise en page pour permettre au développeur de créer n'importe quelle disposition avec une combinaison de ces dispositions de base:
  - **FrameLayout** (un seul élément)
  - **LinearLayout** (plusieurs éléments placés horizontalement ou verticalement sans ascenseurs)
  - **TableLayout** (plusieurs éléments en tableau sans ascenseurs)
  - **RelativeLayout** (plusieurs éléments placés relativement les uns aux autres)
  - **ConstraintLayout** (plusieurs éléments placés selon des "**contraintes**" de positionnement, similaires aux règles du RelativeLayout
  - …

FrameLayout



- Ne contient qu'un seul élément (si on en met plusieurs, ils se superposent)

- Propriétés supplémentaires :

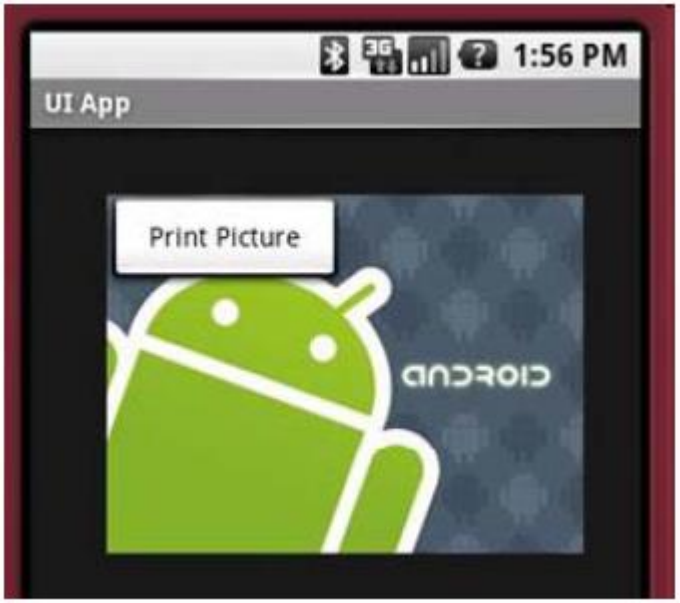
- **Contenu**
  - *android:foreground Pour définir une couleur ou une* image.
  - *android:foregroundGravity Pour positionner l'image*

FrameLayout

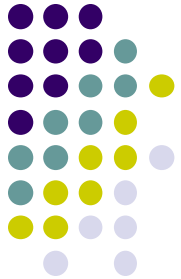
```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_x="40px"
   android:layout_y="35px">

   <ImageView
      android:id="@+id/imageView"
      android:layout_width="349dp"
      android:layout_height="317dp"
      android:src="@drawable/ic_launcher" />
</FrameLayout>
```

FrameLayout

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_x="40px"
    android:layout_y="35px">

    <ImageView
    android:src = "@drawable/androidlogo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

    <Button
    android:layout_width="124px"
    android:layout_height="wrap_content"
    android:text="Print Picture"/>

</FrameLayout>
```



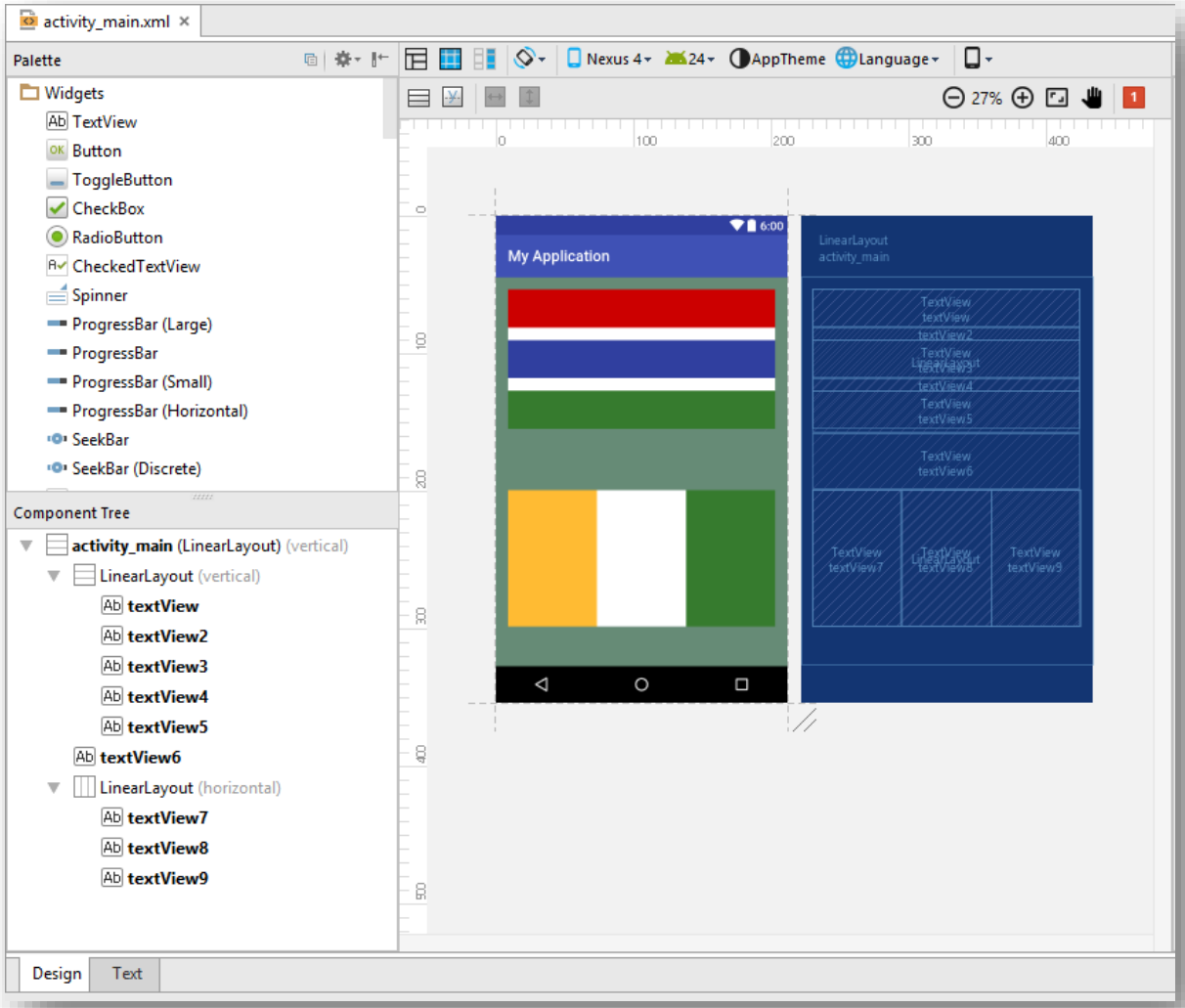
LinearLayout



- **LinearLayout** est un *layout* de vues qui aligne tous les éléments à l'intérieur dans une seule direction, verticalement ou horizontalement. Il est possible de spécifier la direction de la mise en page avec l'attribut **android:d'orientation**.

Attribut	Description
android:orientation	Définit la direction dans laquelle les éléments vont s'étendre. Une valeur « horizontal » ou « vertical » les alignera verticalement ou horizontalement
android:layout_width	Définissent respectivement la largeur et la hauteur du layout. La valeur wrap_content signifie qu'il ne prendra que l'espace nécessaire pour afficher toutes les vues à l'intérieur. La valeur fill_parent va essayer de prendre toute l'espace disponible sur le parent.
android:layout_height	
android:layout_weight	Ce paramètre donne du poids au layout par rapport aux autres vues du parent. Plus la valeur est grande plus le layout courant prendra de l'espace au détriment des autres vues.

LinearLayout pour les drapeaux Gambien et Ivoirien



Following code uses the linear layouts to draw the Gambia and Ivory Coast flags.

LinearLayout pou
Gambiens et Ivoi

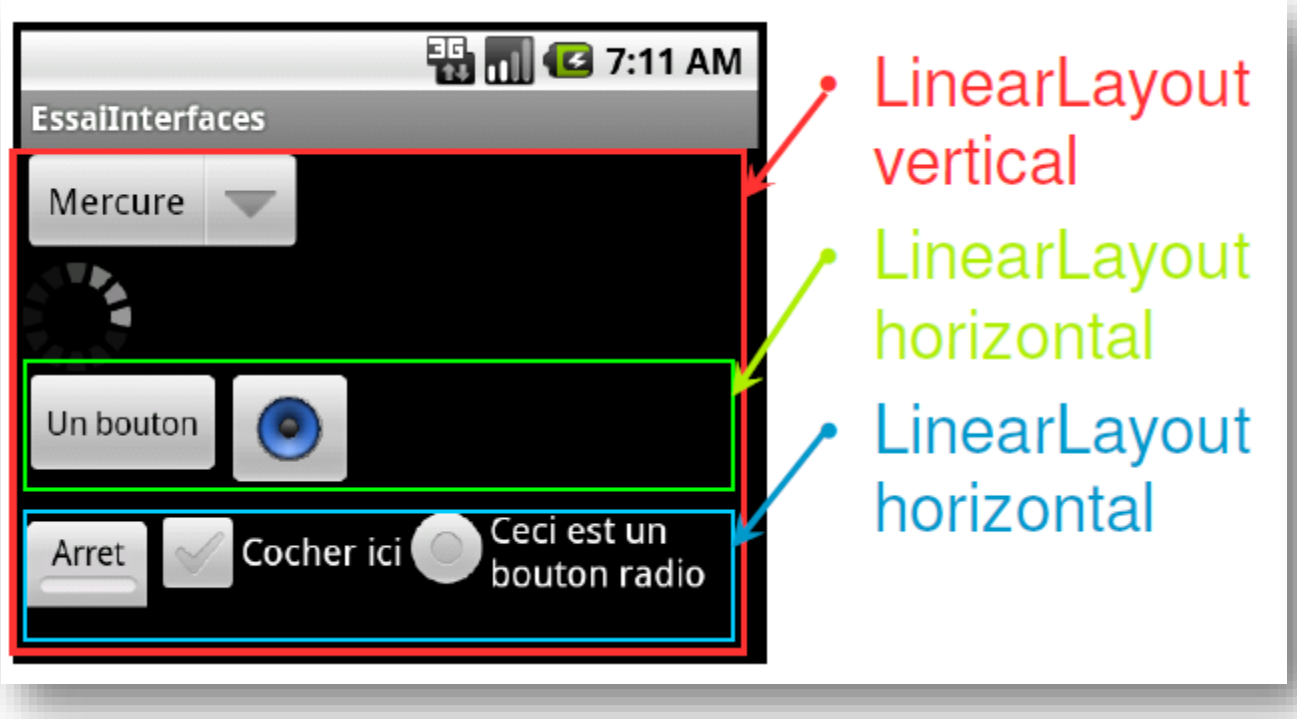
LinearLayout

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.toh.myapplication.MainActivity"
    android:background="@color/windowBackCol"
    android:orientation="vertical"
    android:weightSum="1">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="90dp"
        android:layout_weight="0.31">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/textView"
            android:background="@android:color/holo_red_dark"
            android:height="50dp" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/textView2"
            android:background="@android:color/background_light" />
```

LinearLayout

```xml
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/textView3"
            android:height="50dp"
            android:background="@color/colorPrimaryDark" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/textView4"
            android:background="@android:color/background_light" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/textView5"
            android:background="@color/green"
            android:height="50dp" />

    </LinearLayout>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:id="@+id/textView6"
        android:height="50dp"
        android:layout_weight="0.23" />

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="67dp"
        android:layout_weight="0.35">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView7"
            android:layout_weight="1"
            android:background="@android:color/holo_orange_light" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView8"
            android:layout_weight="1"
            tools:background="@android:color/background_light" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/textView9"
            android:layout_weight="1"
            android:background="@color/green" />
    </LinearLayout>
</LinearLayout>
```

Exemple avec LinearLayout



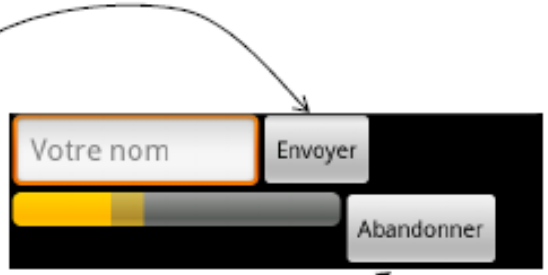
RelativeLayout

- Sur cette méthode chaque élément est sensé définir sa position relativement aux autres éléments dans le même layout.
- Chaque élément défini sa position en fonction des éléments qui ont déjà été définies et les éléments suivants peuvent définir leur position en fonction du même élément. Les propriétés utilisées sont :

Propriété	Signification
android:layout_alignParentTop	Aligner sur le haut du layout
android:layout_alignParentEnd	Aligner à gauche du layout
android:layout_alignParentStart	Aligner à droite du parent
android:layout_alignBottom	Aligner en dessous du widget indiqué
android:layout_toStartOf	Aligner avant le widget indiqué
android:layout_alignTop	Aligner au-dessus du Widget indiqué

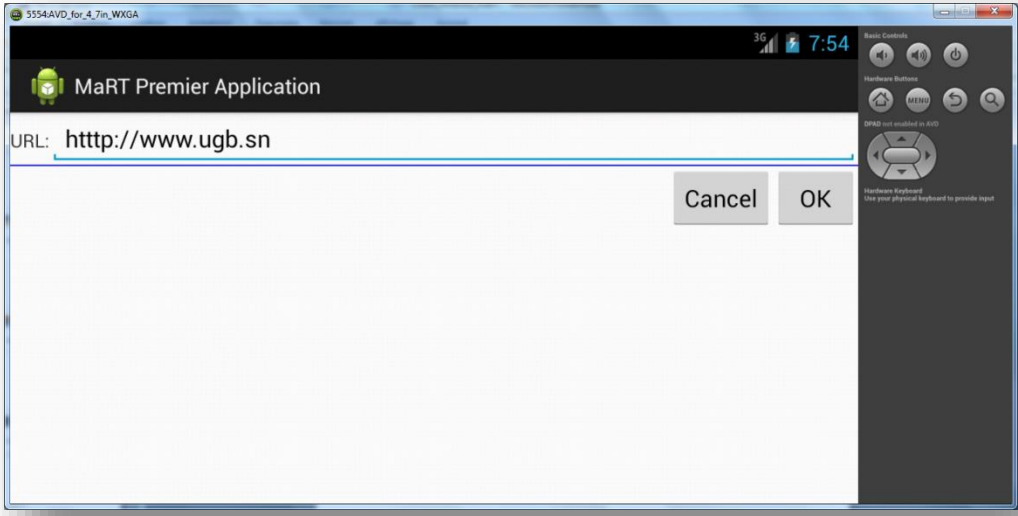
Exemple avec RelativeLayout

```
<RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_height="fill_parent"
android:layout_width="fill_parent">
    <EditText android:id="@+id/nom"
    ..."
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    />
    <Button android:id="@+id/envoyer"
    android:layout_toRightOf="@+id/nom"
    ...
    />
    <ProgressBar android:id="@+id/attente"
    android:layout_below="@+id/nom"
    ...
    />
    <Button android:id="@+id/annuler"
    android:layout_toRightOf="@+id/attente"
    android:layout_below="@+id/nom"
    ...
    />
</RelativeLayout>
```



Exemple avec RelativeLayout

```
<RelativeLayout
        android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:padding="5px">
         <TextView android:id="@+id/label"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="URL:"
                  android:paddingTop="15px"/>
         <EditText
                  android:id="@+id/entry"
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:layout_toRightOf="@id/label"
                  android:layout_alignBaseline="@id/label"/>
         <Button
                  android:id="@+id/ok"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_below="@id/entry"
                  android:layout_alignRight="@id/entry"
                  android:text="OK" />
         <Button
                  android:id="@+id/cancel"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_toLeftOf="@id/ok"
                  android:layout_alignTop="@id/ok"
                  android:text="Cancel" />
</RelativeLayout>
```



ConstraintLayout

- Pour placer les éléments selon des “**contraintes**” de positionnement, similaires aux règles du RelativeLayout (*layout_alignTop*, *layout_toRightOf* etc.).

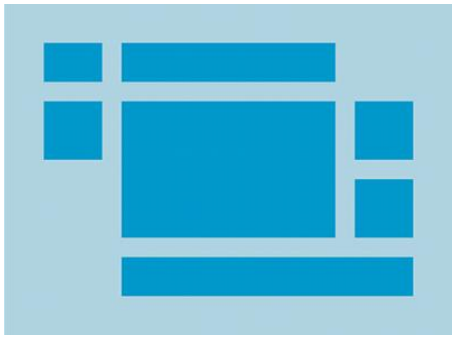
- Ces contraintes sont des attributs de widgets qui prennent la forme suivante dans le XML : **layout_constraint(X)_to(Y)of**

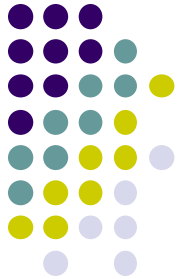
  Avec X le côté du widget où l’on place la contrainte, et Y le côté du widget de destination.

  Par exemple, si je veux aligner la droite de mon widget à la gauche d’un autre widget, j’utiliserais la contrainte suivante : **layout_constraintLeft_toRightOf**

```
app:layout_constraintLeft_toLeftOf="@+id/auth_background_view"
app:layout_constraintTop_toBottomOf="@+id/login_edittext"
```

TableLayout



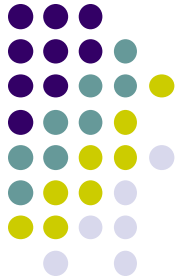


- Pour placer des éléments en tableau sans ascenseurs (pour en avoir le mettre dans un ScrollView et/ou un HorizontalScrollView).

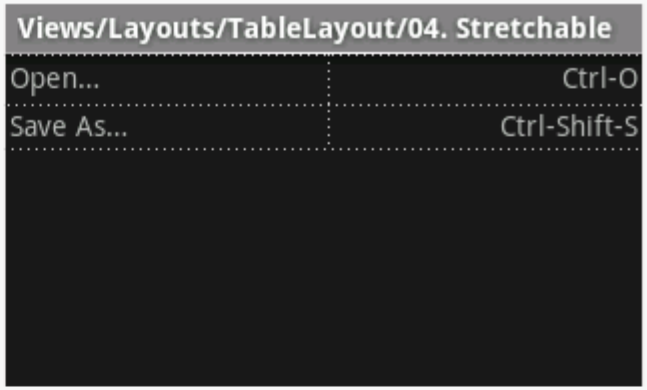
https://developer.android.com/reference/android/widget/TableLayout.html

- Propriétés supplémentaires :
  - android:collapseColumns Pour définir les numéros de colonnes à cacher
  - android:shrinkColumns Pour définir les numéros de colonnes qui peuvent être rétrécies en fonction de la place disponible
  - android:stretchColumns Pour définir les numéros de colonnes qui peuvent être agrandies en fonction de leur contenu
  - Chaque élément ajouté dans un TableLayout indiquera le nombre de colonnes qu'il occupe en mettant dans ses propriétés : android:layout_span (par défaut 1)

Exemple avec TableLayout

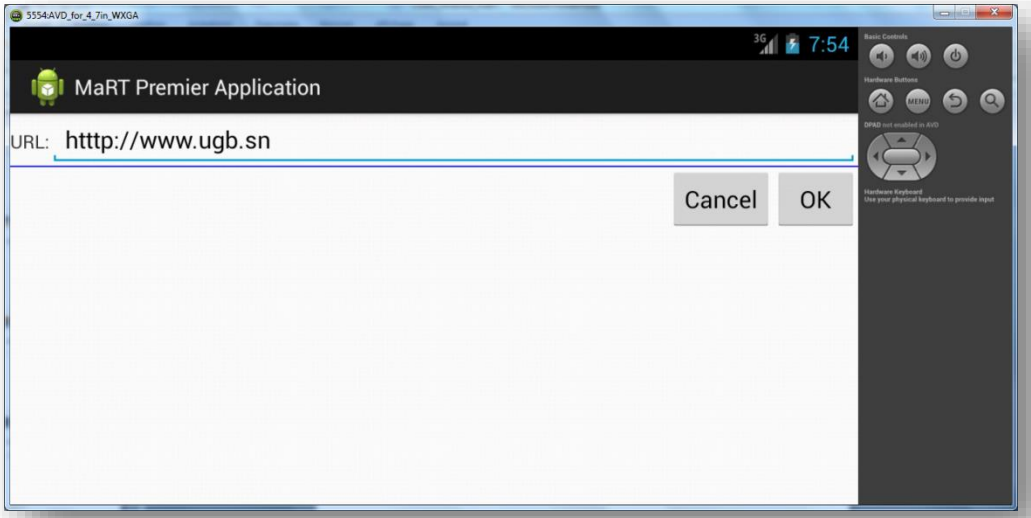


```xml
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="@string/table_layout_4_open"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_open_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:text="@string/table_layout_4_save"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_save_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>
</TableLayout>
```

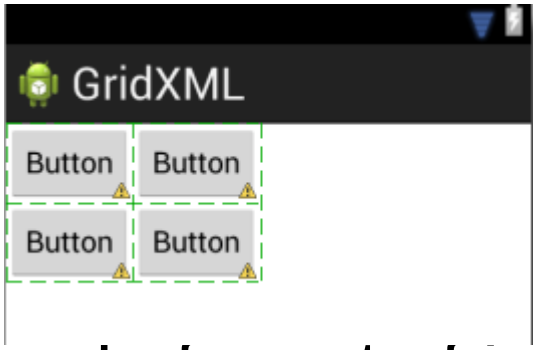


Exemple avec TableLayout

```
<TableLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="1">
        <TableRow>
                <TextView
                  android:text="URL:" />
                <EditText android:id="@+id/entry"
                 android:layout_span="3"/>
        </TableRow>
        <View
                android:layout_height="2px"
                android:background="#0000FF" />
        <TableRow>
                <Button android:id="@+id/cancel"
                        android:layout_column="2"
                        android:text="Cancel" />
                <Button android:id="@+id/ok"
                        android:text="OK" />
      </TableRow>
</TableLayout>
```

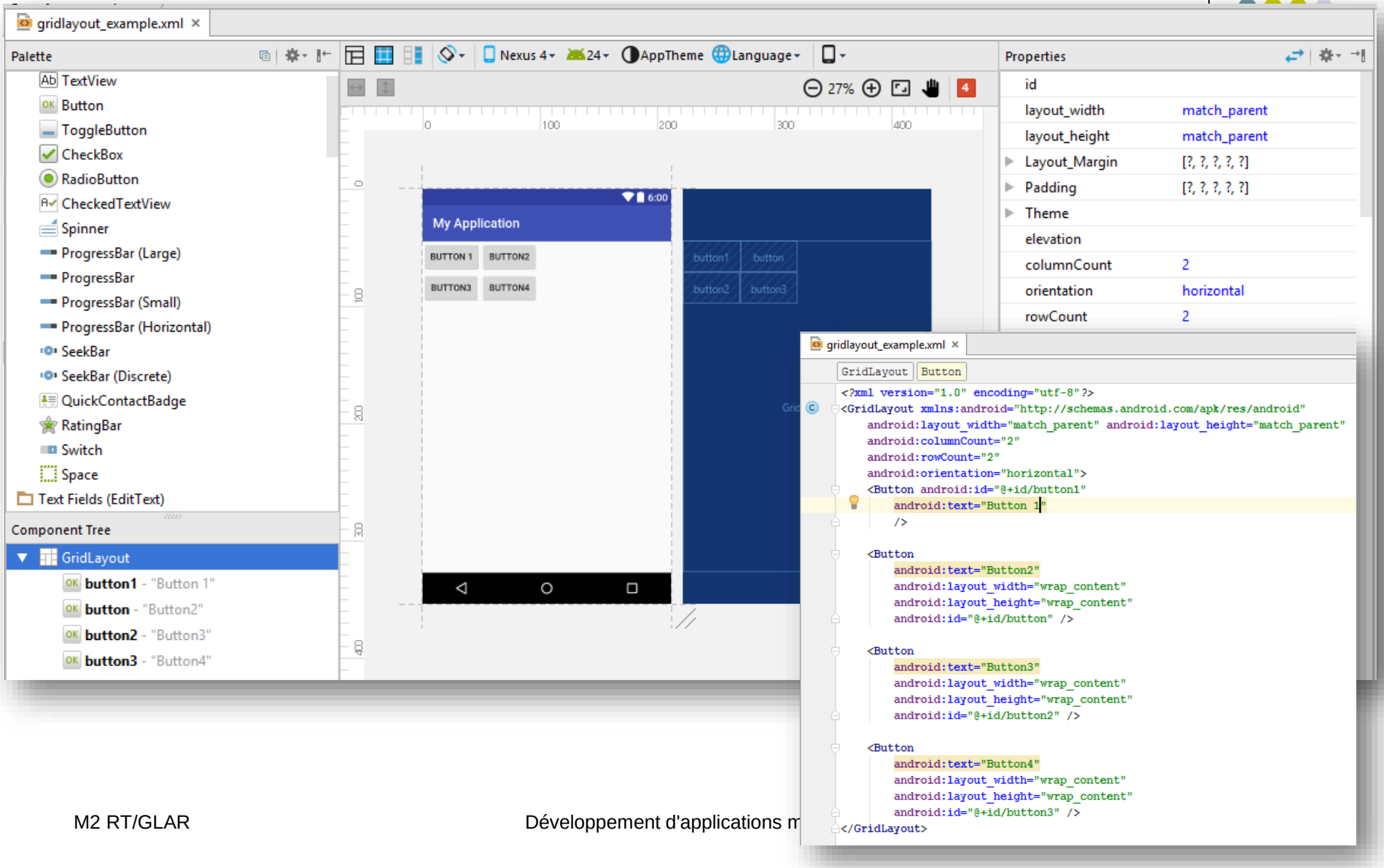


GridLayout



- The grid is composed of a set of infinitely thin lines that separate the viewing area into cells. Throughout the API, grid lines are referenced by grid indices.
- A grid with N columns has N + 1 grid indices that run from 0 through N inclusive.
- Regardless of how GridLayout is configured, grid index 0 is fixed to the leading edge of the container and grid index N is fixed to its trailing edge (after padding is taken into account).

GridLayout: exemple



Développement d'applications mobiles

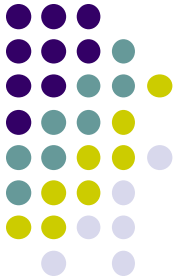
Android Widgets

M2 RT/GLAR

PARTIE 4

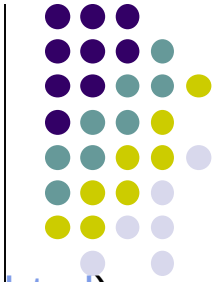
Pr. El hadji M. NGUER

Widgets



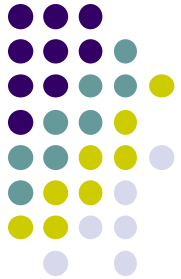
- Les widgets sont les contrôles utilisés pour interagir avec l'utilisateur ou lui afficher des informations.
- Dans cette palette nous avons des widgets pour l'affichage de texte, des images, pour créer des boutons, des champs de saisie de texte ou les autres éléments communs des formulaires.
- Pour les boutons, sont des éléments graphiques sur lesquels l'utilisateur peut appuyer ou cliquer pour obtenir une réaction.

Main widgets offered by Android



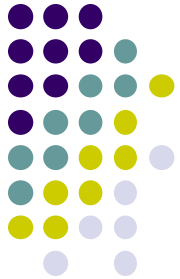
- **Button**(http://developer.android.com/reference/android/widget/Button.html) : Un bouton cliquable.
- **CheckBox**(http://developer.android.com/reference/android/widget/CheckBox.html) : Une checkbox.
- **EditText**(http://developer.android.com/reference/android/widget/EditText.html) : Un champ de texte éditable.
- **DatePicker**(http://developer.android.com/reference/android/widget/DatePicker.html) : Sélection de dates.
- **RadioButton**(http://developer.android.com/reference/android/widget/RadioButton.html) : Représente les boutons radios.
- **Toast**(http://developer.android.com/reference/android/widget/Toast.html) : Un pop up message qui s'affiche sur l'écran.
- **ImageButton**(http://developer.android.com/reference/android/widget/ImageButton.html): Une image qui se comporte comme un bouton.
- **SlidingDrawer :** Un élément qui se présente sous forme d'un tiroir qu'on ouvre et ferme.

Propriété communes aux éléments d'interface (conteneurs et widgets)



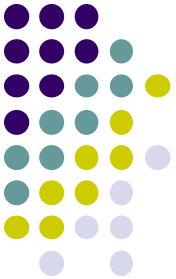
- ***Identifiant :*** Un identifiant peut être associé à chaque élément décrit dans un fichier XML, cet identifiant permet d'accéder à l'objet créé dans le code ou de le référencer dans d'autres fichiers XML. Les éléments ne devant pas être référencés peuvent ne pas avoir d'identifiant.
- **android:id**="@+id/*mon_ident"* permettra de retrouver cet élément par findViewById(R.id.*mon_ident).*
- Méthode correspondante : **setId(int)**

Propriété communes aux éléments d'interface (conteneurs et widgets)

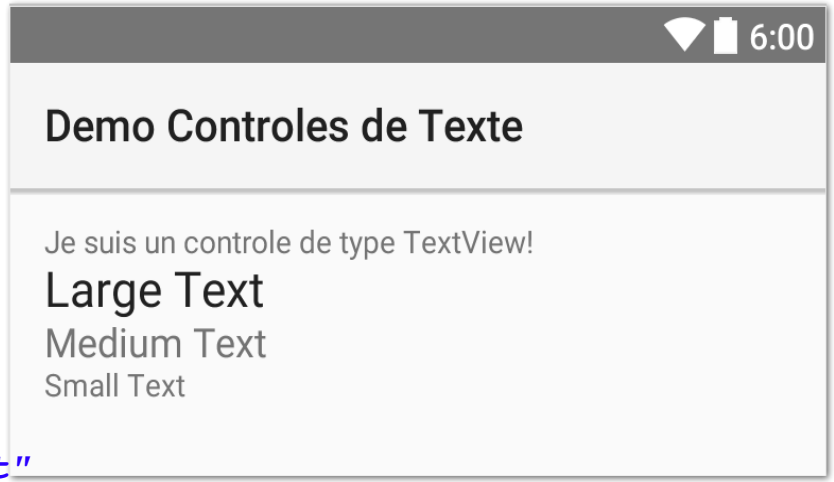


- ***Visibilité***
  - android:visibility : Rend l'élément visible, invisible ou absent (avec invisible la place est conservée, avec absent la place n'est pas conservée .
- ***Fond***
  - android:background couleur ou une image de fond
- ***Taille***
  - android:minHeight et android:minWidth dimensions minimales
- ***Placement des éléments contenus (défini pour chaque élément)***
  - android:layout_height et android:layout_width place prise par l'élément dans le conteneur :
    - FILL_PARENT rempli toute la place
    - WRAP_CONTENT occupe la place nécessaire
  - android:layout_gravity positionnement de l'élément dans le conteneur top, bottom, left, right, center_vertical, fill_vertical, center_horizontal, fill_horizontal, center, fill

Les labels de texte avec XML (dans activity_main.xml)



- TextView est normalement utilisé pour afficher un texte
- propriétés souvent utilisées:
  - **TextView,**
  - **EditText,**
  - **AutoCompleteTextView,**
  - **MultiCompleteTextView.**

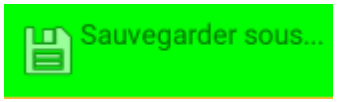



```
<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
android:text="@string/hello_world" />
```

Text-based widgets: TextView

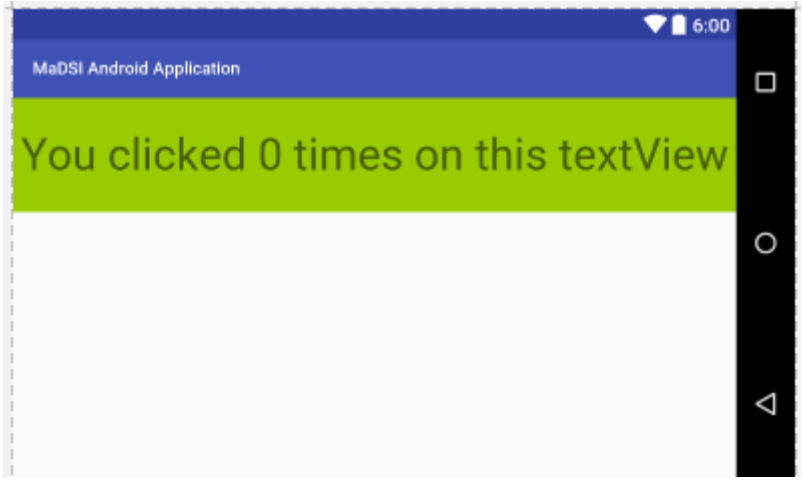
- Pour manipuler d'un TextView il faut récupérer la référence de l'objet avec la méthode findViewByID de la classe Activity.
- La méthode **setText** de TextView permet de modifier le texte présenté tandis que la méthode **getText()** permet de récupérer le contenu actuelle du texte.

drawableStart	@android:drawable/ic_menu_save



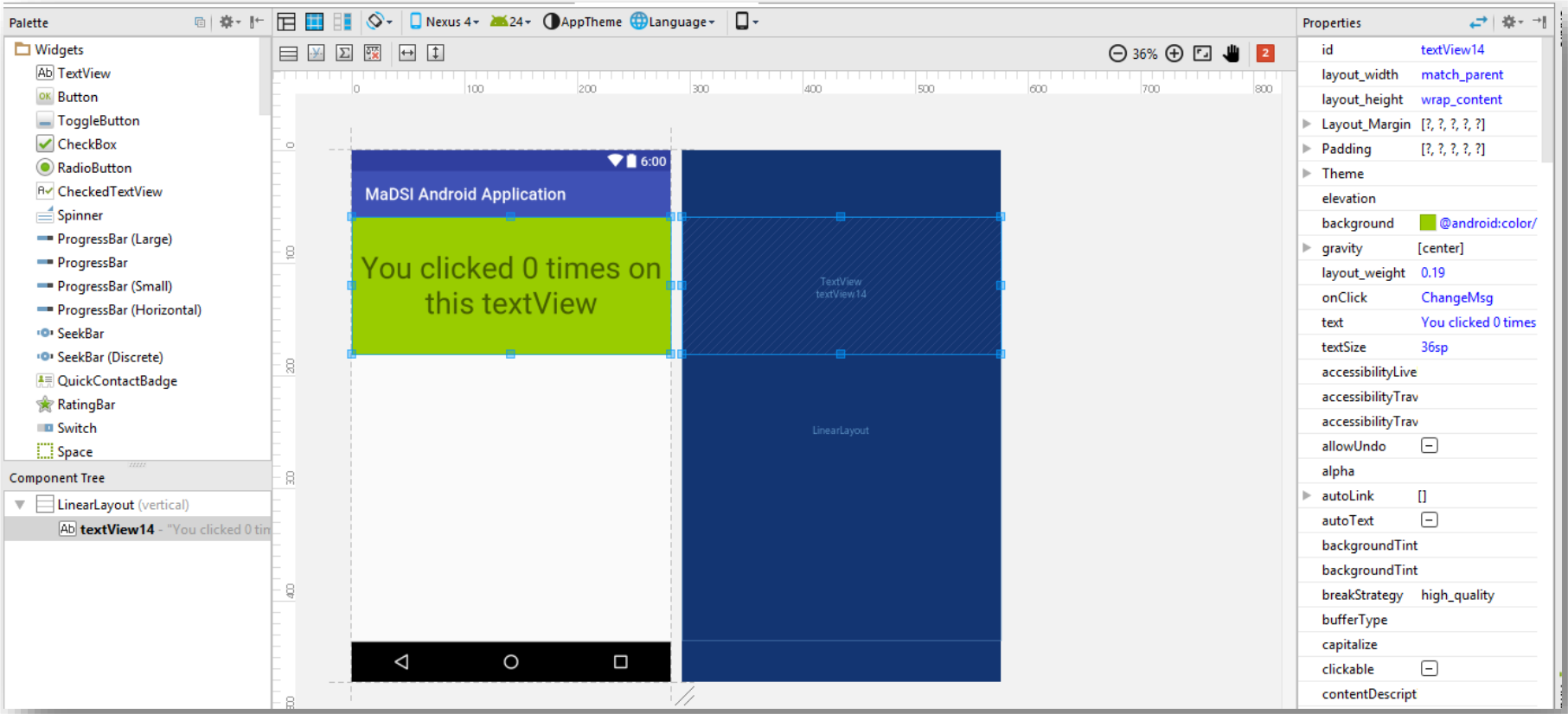
Text-based widgets: TextView

- Considérons l'application suivante qui donne un exemple d'utilisation du textView. Elle permet de compter le nombre de clicks de l'utilisateur sur le textView



Text-based widgets: TextView

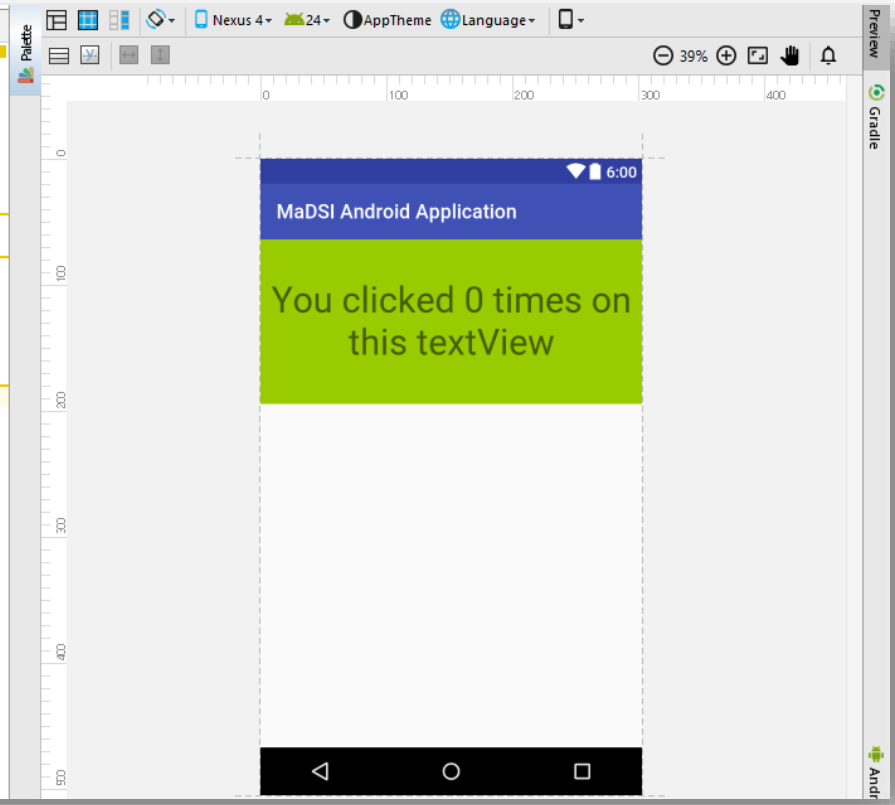
- Considérons l'application suivante qui donne un exemple d'utilisation du textView.



Text-based widgets: TextView

- Considérons l'application suivante qui donne un exemple d'utilisation du textView.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="1">

    <TextView
        android:text="You clicked 0 times on this textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView14"
        android:layout_weight="0.19"
        android:background="@android:color/holo_green_light"
        android:gravity="center"
        android:textSize="36sp"
        android:onClick="ChangeMsg" />
</LinearLayout>
```



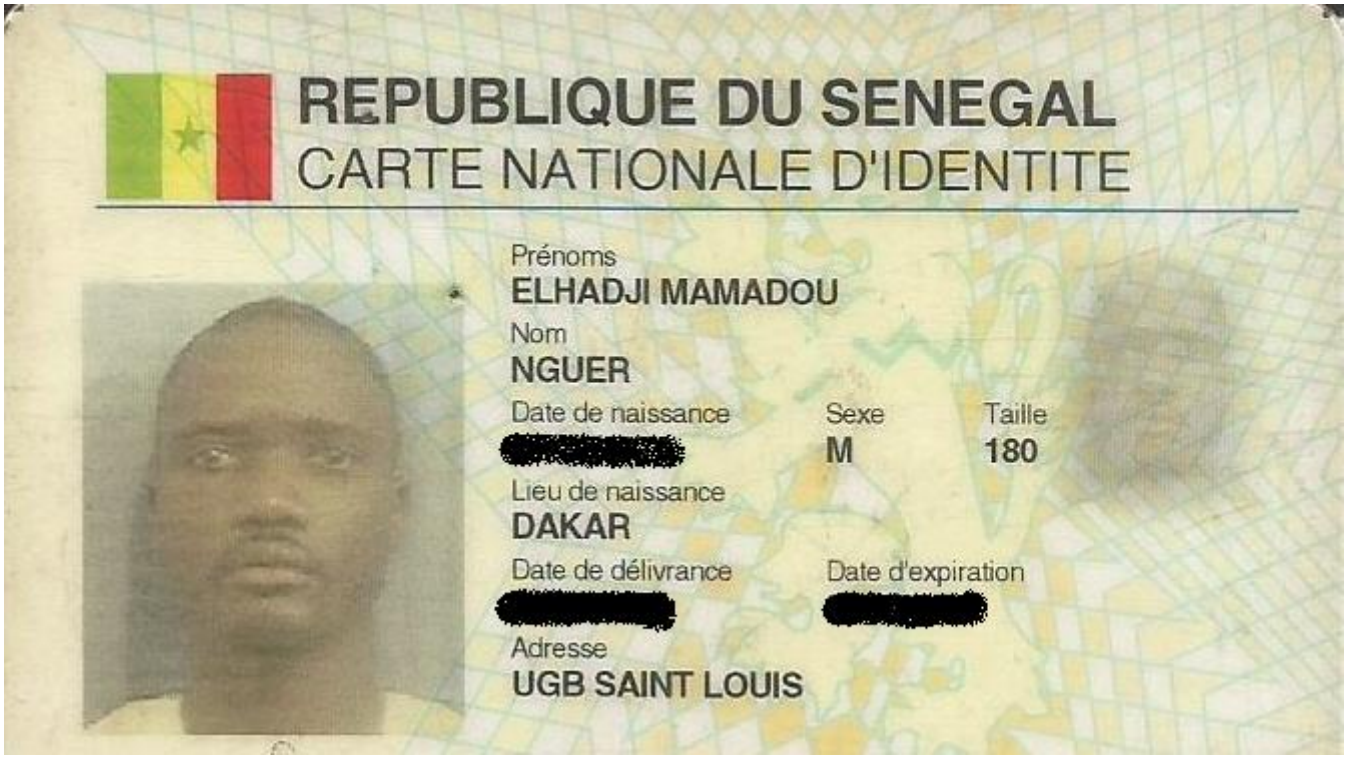
Text-based widgets: TextView

- Considérons l'application suivante qui donne un exemple d'utilisation du textView.

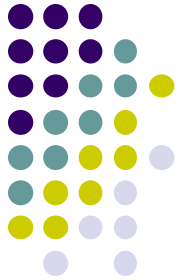
```
1  package com.example.toh.myfirstapplication;
2
3  import android.support.v7.app.AppCompatActivity;
4  import android.os.Bundle;
5  import android.view.View;
6  import android.widget.Button;
7  import android.widget.TextView;
8  import android.widget.Toast;
9
10 public class MainActivity extends AppCompatActivity {
11     TextView myTextView;
12     int counter=1;
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.textview_example);
17         myTextView=(TextView) findViewById(R.id.textView14);
18         myTextView.setOnClickListener(new View.OnClickListener() {
19             @Override
20             public void onClick(View v) {
21                 myTextView.setText("You clicked "+(counter++)+" times on this textView");
22             }
23         });
24     }
25 }
```

Exercice

- Créer une fenêtre similaire à celle ci-dessous



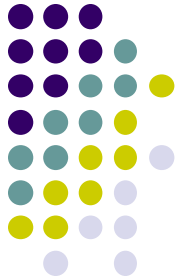
Les zones de texte : EditText



- EditText est normalement utilisé pour saisir du texte
- Hérite de **TextView**

- Propriétés:
  - android:autoText
  - android:capitalize
  - android:digits
  - android:singleLine
  - android:numeric
  - android:phoneNumber
  - android:password
  - android:inputType («number», «phone»,...)
  - android:hint

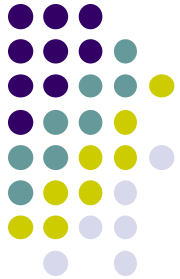
Les zones de texte : EditText



- Le widget EditText permet de donner des champs dans lesquelles l'utilisateur peut insérer du texte. Par exemple, l'application SMS utilise ce widget pour récupérer le contenu du message.
- Ce champ offre des fonctionnalités très intéressantes à travers sib attribut **android:inputType**. Cet attribut permettre aussi bien de contrôler ce que l'utilisateur peut entrer ou bien la correction automatique des erreurs orthographiques.

- Propriétés:
  - android:autoText
  - android:capitalize
  - android:digits
  - android:singleLine
  - android:numeric
  - android:phoneNumber
  - android:password
  - android:inputType («number», «phone»,...)
  - android:hint

Les zones de texte : EditText



- XML

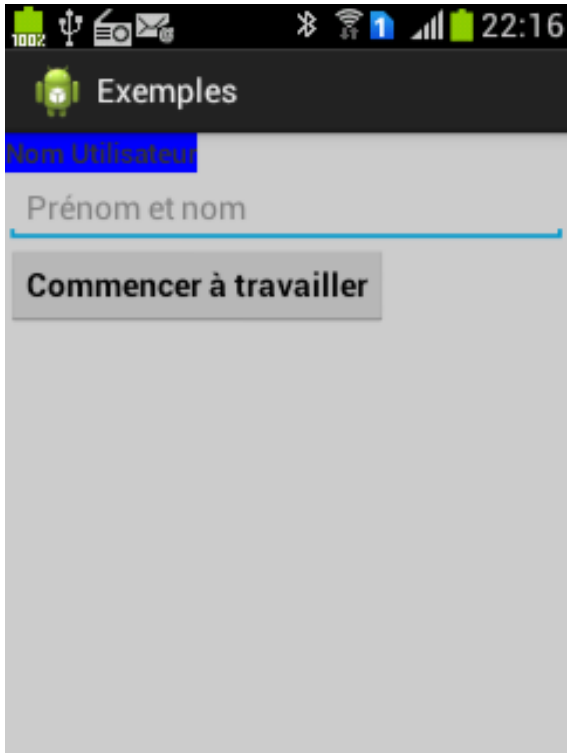
```
<EditText
        android:id="@+id/EditText01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="" >
</EditText>
```

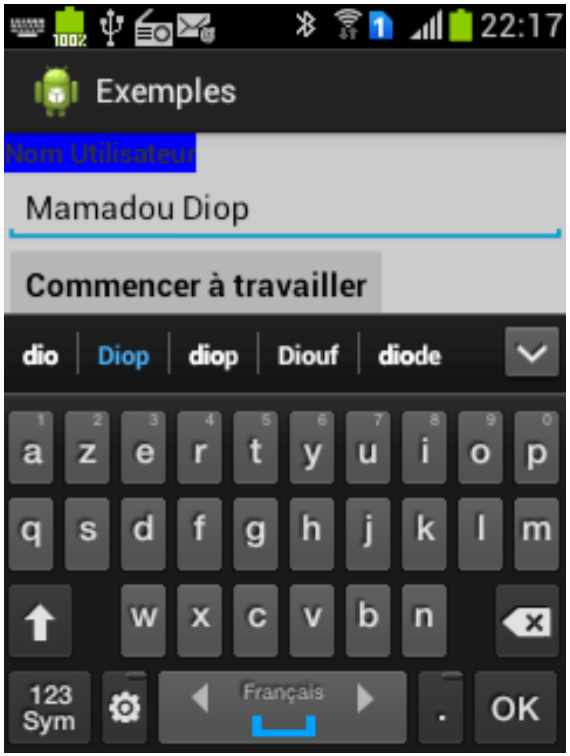
- Java

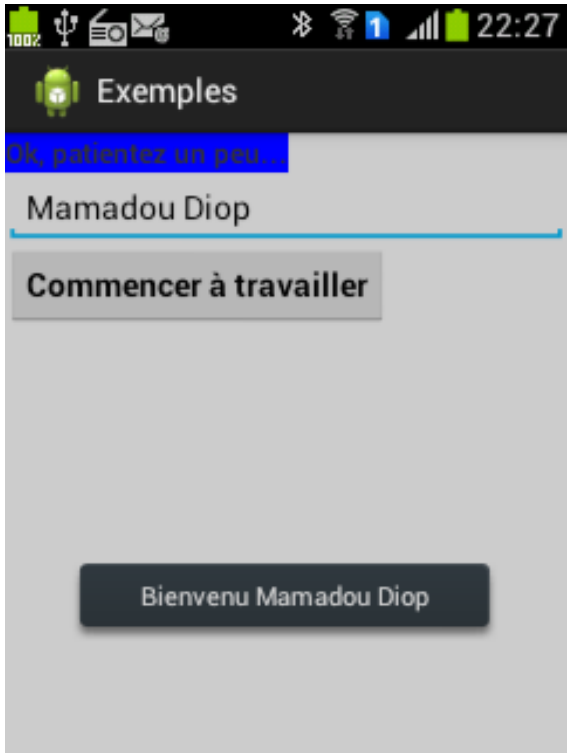
```
EditText edit = new EditText(this);
edit.setText("Edite moi s'il te plait ");
myLayout.addView(edit);
```

Zones de texte: exemple

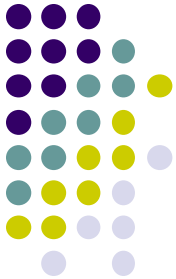
Dans ce petit exemple, nous utiliserons un LinearLayout détenant un label (TexView), un textBox(EditText) et un bouton. Nous allons utiliser la vue comme une sorte d'écran de connexion simplifiée.







Text-based widgets: EditText



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffcccccc"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/labelUserName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:text="Nom Utilisateur"
        android:textSize="16sp"
        android:textStyle="bold" />
```

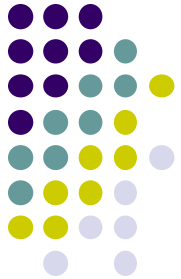
```
    <EditText
        android:id="@+id/txtUserName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:autoText="true"
        android:capitalize="words"
        android:ems="10"
        android:hint="Prénom et nom"
        android:inputType="textPersonName"
        android:textSize="18sp" />

    <Button
        android:id="@+id/bouttonCommencer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Commencer à travailler"
        android:textSize="14px"
        android:textStyle="bold" />
</LinearLayout>
```

Text-based widgets: EditText

```
1  package com.example.exemples;
2
3  import android.os.Bundle;
12
13 public class MainActivity extends Activity implements OnClickListener {
14     Button monButton;
15     EditText txtUserName;
16     TextView labelUserName;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22         labelUserName = (TextView) findViewById(R.id.labelUserName);
23         txtUserName = (EditText) findViewById(R.id.txtUserName);
24         monButton = (Button) findViewById(R.id.bouttonCommencer);
25         monButton.setOnClickListener(this);
26     }
```

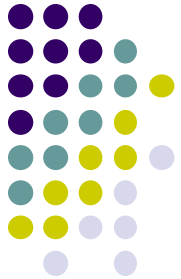
```
27     @Override
28     public void onClick(View v) {
29         // TODO Auto-generated method stub
30         if (v.getId() == monButton.getId()) {
31             String userName = txtUserName.getText().toString();
32             if (userName.compareTo("Mamadou Diop") == 0) {
33                 labelUserName.setText("Ok, patientez un peu...");
34                 Toast.makeText(getApplicationContext(), "Bienvenu " + userName,
35                         Toast.LENGTH_LONG).show();
36             }
37             Toast.makeText(getApplicationContext(), "Bienvenu " + userName,
38                     Toast.LENGTH_LONG).show();
39         }
40     }
```

Button



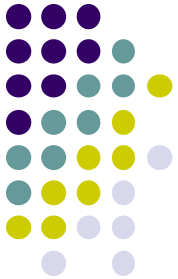
- Le boutons sont des contrôles qui communiquent les évènements lorsque l'utilisateur les touche avec le tactile.
- Un bouton peut contenir du texte, une image ou les deux.
- L'attribut **android:text** permet d'ajouter du texte sur l'image alors que les attributs android:drawable* permettent d'insérer une icône et sur le bouton.

Les boutons : Button



- Button : Mêmes paramètres que TextView
- ImageButton : Mêmes paramètres que ImageView càd :
  - android:src="couleur" pour définir une couleur ou une image
  - android:adjustViewBounds Pour indiquer si la taille du bouton doit ou pas être ajustée à celle de l'image
  - android:baselineAlignBottom Pour indiquer que l'image est placée ou pas en bas de la zone
  - android:cropToPadding Pour indiquer si l'image sera coupée ou pas si elle est plus grande que la taille disponible
  - android:scaleType="s" (où s peut prendre les valeurs : matrix, fitXY, fitStart, fitCenter, fitEnd, center, centerCrop, centerInside) permet de redimensionner ou pas l'image à la taille disponible et/ou de la déformer.
  - android:maxHeight Pour définir la hauteur disponible
  - android:maxWidth Pour définir la largeur disponible
  - android:tint Pour définir une couleur qui teinte l'image

Les boutons

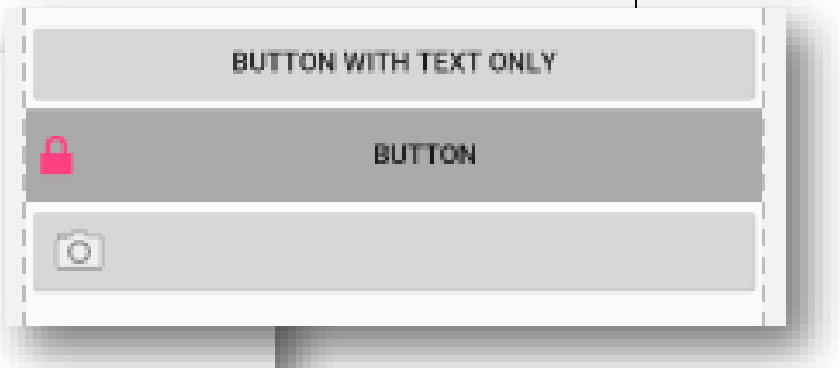


```
<Button
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Clique sur moi " >
</Button>
```

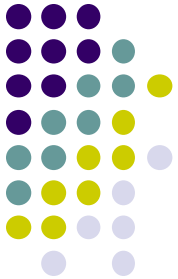
```
Button b = (Button) findViewById(R.id.Button01);
b.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
Toast.makeText(v.getContext(), "Stop ", Toast.LENGTH_LONG).show();
}
});
myLayout.addView(b);
```

Button

```
<Button
    android:text="Button with text only"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/button4" />

<Button
    android:text="Button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/button5"
    android:drawableLeft="@android:drawable/ic_lock_lock"
    android:background="@android:color/darker_gray"
    tools:ignore="RtlHardcoded" />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/button6"
    android:drawableLeft="@android:drawable/ic_menu_camera" />
```



ImageButton



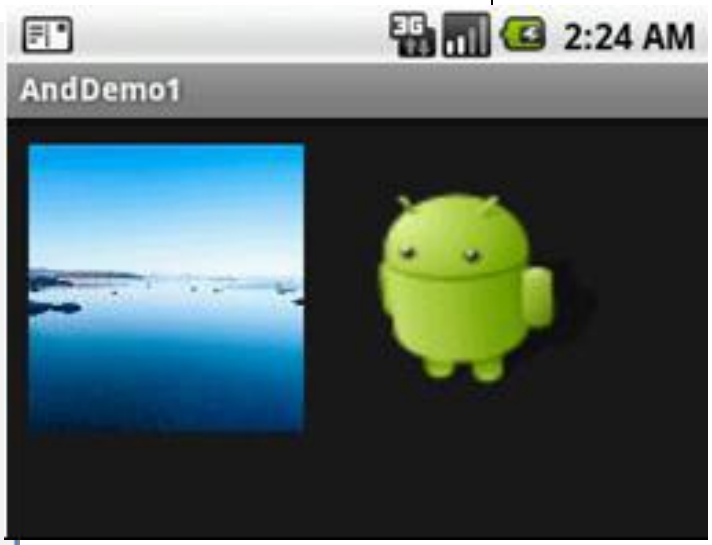
- Bouton normal, mais avec une image. Mêmes paramètres XML que ImageView
- Exemple de fichier XML :

```
<ImageButton android:id="@+id/boutonloupe"
android:src="@drawable/loupe"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

- **Méthodes de la classe ImageButton**
  - android.widget.ImageButton
  - ImageButton hérite de ImageView il a donc les mêmes sauf le constructeur :
- ***Construction:*** ImageButton(Context) le paramètre est généralement l'activité elle-même

ImageButton

```
. . .
<ImageButton
      android:id="@+id/myImageBtn1"
      android:background="@drawable/default_wallpaper"
      android:layout_width="125px"
      android:layout_height="131px"
>
</ImageButton>

<ImageView
      android:id="@+id/myImageView1"
      android:background="@drawable/ic_launcher_android"
      android:layout_width="108px"
      android:layout_height="90px"
>
</ImageView>
```



Les éléments à deux états

Ils ont les mêmes paramètres que TextView auxquels vient s'ajouter la définition de l'état initial :android:checked="b" où b vaut true ou false Pour définir l'état initial

- CheckBox 

- RadioButton 

- ToggleButton 

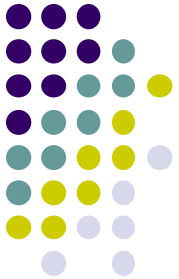
  - android:disabledAlpha pour définir la transparence appliquée lorsque le bouton est inactif
  - android:textOff Pour définir le texte quand le bouton n'est pas allumé
  - android:textOn Pour définir le texte quand le bouton est allumé

Les CheckBox

- Hérite de **CompoundButton**
- Possible d'alterer la valeur depuis le code:
  - **isChecked()**
  - **setChecked()**
  - **toggle()**
  - Possible d'enregistrer un event listener **OnCheckedChangeListener**

```
<CheckBox
    android:id="@+id/check"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This checkbox is: unchecked" />
```

Exemple CheckBox



```
public class CheckBoxDemo extends Activity implements CompoundButton.OnCheckedChangeListener {
    CheckBox cb;

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);

         cb=(CheckBox)findViewById(R.id.check);
        cb.setOnCheckedChangeListener(this);
    }

    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            cb.setText("Ce checkbox est: coché");
        }
        else {
            cb.setText("Ce checkbox n'est pas coché");
        }
    }
}
```

CheckBox: exemple

- Un exemple d'utilisation d'un contrôle checkbox à l'intérieur de votre activité est le suivant :

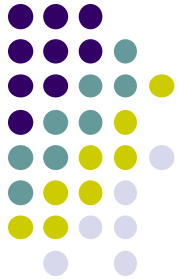
Basic Widgets: CheckBox

activity_main.xml

```
1  <LinearLayout
2      xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:id="@+id/LinearLayout1"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical"
8      tools:context=".MainActivity" >
9
10     <TextView
11         android:id="@+id/textView1"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:layout_gravity="center"
15         android:layout_marginBottom="7dp"
16         android:layout_marginTop="10dp"
17         android:gravity="center"
18         android:text="@string/txtBienvenue"
19         android:textColor="#FF0000"
20         android:textSize="22sp"
21         android:textStyle="bold"
22         android:typeface="monospace" />
23
```

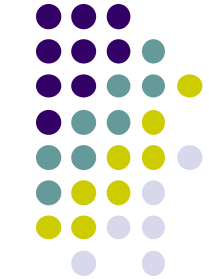
```
24     <LinearLayout
25         android:id="@+id/monPanneau"
26         android:layout_margin="10dp"
27         android:background="#FFFF00"
28         android:orientation="vertical" >
29         <CheckBox
30             android:id="@+id/dejeuner"
31             android:layout_width="wrap_content"
32             android:layout_height="wrap_content"
33             android:layout_margin="10dp"
34             android:text="@string/txtDejeuner" />
35         <CheckBox
36             android:id="@+id/dessert"
37             android:layout_width="wrap_content"
38             android:layout_height="wrap_content"
39             android:layout_margin="10dp"
40             android:text="@string/txtDessert" />
41         <CheckBox
42             android:id="@+id/the"
43             android:layout_width="wrap_content"
44             android:layout_height="wrap_content"
45             android:layout_margin="10dp"
46             android:text="@string/txtThe" />
47         <CheckBox
48             android:id="@+id/boisson"
49             android:layout_width="wrap_content"
50             android:layout_height="wrap_content"
51             android:layout_margin="10dp"
52             android:text="@string/txtBoisson" />
53     </LinearLayout>
```

Basic Widgets: CheckBox



```
55      </LinearLayout>
56      <RelativeLayout
57          android:id="@+id/layout2"
58          android:layout_width="wrap_content"
59          android:layout_height="wrap_content" >
60
61          <Button
62              android:id="@+id/button1"
63              android:layout_width="wrap_content"
64              android:layout_height="wrap_content"
65              android:layout_alignParentLeft="true"
66              android:layout_alignParentTop="true"
67              android:layout_marginBottom="17dp"
68              android:layout_marginTop="16dp"
69              android:text="@string/txtValider"
70              android:textStyle="bold"
71              android:typeface="monospace" />
72
73          <Button
74              android:id="@+id/button2"
75              android:layout_width="wrap_content"
76              android:layout_height="wrap_content"
77              android:layout_alignBaseline="@+id/button1"
78              android:layout_alignParentTop="true"
79              android:layout_toRightOf="@+id/button1"
80              android:text="@string/quitter" />
81
82      </RelativeLayout>
83
84  </LinearLayout>
```

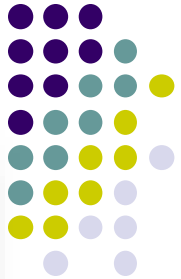
Basic Widgets: CheckBox



activity_main.xml | MainActivity.java

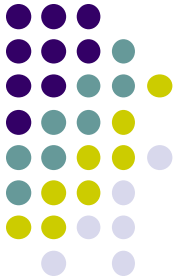
```
1  package com.example.exemple1checkbox;
2
3  import android.os.Bundle;
4  import android.app.Activity;
5  import android.view.Menu;
6  import android.view.View;
7  import android.view.View.OnClickListener;
8  import android.widget.Button;
9  import android.widget.CheckBox;
10 import android.widget.Toast;
11
12 public class MainActivity extends Activity implements OnClickListener {
13     Button valider;
14     Button quitter;
15     CheckBox dejeuner;
16     CheckBox dessert;
17     CheckBox the;
18     CheckBox boisson;
19
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22         super.onCreate(savedInstanceState);
23         setContentView(R.layout.activity_main);
24         valider = (Button) findViewById(R.id.button1);
25         quitter = (Button) findViewById(R.id.button2);
26         dejeuner = (CheckBox) findViewById(R.id.dejeuner);
27         dessert = (CheckBox) findViewById(R.id.dessert);
28         the = (CheckBox) findViewById(R.id.the);
29         boisson = (CheckBox) findViewById(R.id.boisson);
30         valider.setOnClickListener(this);
31         quitter.setOnClickListener(this);
```

Basic Widgets: CheckBox



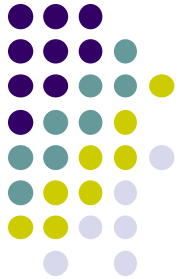
```
32      }
33
34      @Override
35      public boolean onCreateOptionsMenu(Menu menu) {
36          // Inflate the menu; this adds items to the action bar if it is present.
37          getMenuInflater().inflate(R.menu.activity_main, menu);
38          return true;
39      }
40
41      @Override
42      public void onClick(View v) {
43          if (v.getId() == valider.getId()) {
44              String[] commandes = new String[4];
45              int i = 0;
46              if (dejeuner.isChecked())
47                  commandes[i++] = "Déjeuner";
48              if (dessert.isChecked())
49                  commandes[i++] = "Dessert";
50              if (the.isChecked())
51                  commandes[i++] = "Thé";
52              if (boisson.isChecked())
53                  commandes[i++] = "Boisson";
54              if (i == 0) {
55                  Toast.makeText(getApplicationContext(),
56                          "Vous n'avez rien choisi !!!", Toast.LENGTH_LONG)
57                          .show();
58              } else {
59                  String str = "Vous avez commandé:\n";
60                  for (int j = 0; j < commandes.length; j++) {
61                      if (commandes[j] != null)
62                          str += "\t - " + commandes[j] + "\n";
```

Basic Widgets: CheckBox



```
63                  }
64                  Toast.makeText(getApplicationContext(), str, Toast.LENGTH_LONG)
65                          .show();
66              }
67          } else {
68              System.exit(0);
69          }
70      }
71
72  }
```

RadioButton

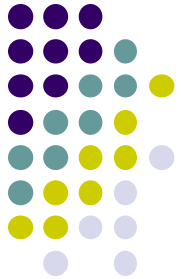


- **RadioButton:** Mêmes paramètres XML que TextView plus android:checked="b" où b vaut true ou false coche ou décoche le bouton au départ
- Exemple de fichier XML :

```
<RadioButton android:id="@+id/gauche"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Left"
android:checked="false"
android:layout_gravity="left"
/>
```

- **Méthodes de la classe RadioButton:** android.widget.RadioButton
- ***Construction:*** RadioButton(Context) le paramètre est généralement l'activité elle-même
- ***Etat***
  - isChecked() renvoie true si la case est cochée
  - setChecked(boolean) coche (true) ou décoche (false) la case
  - toggle() change l'état de la case

RadioButtons



- Une case d'option est un bouton de deux États qui peut être soit cochée ou décochée.
- Si la case d'option est désactivée, l'utilisateur peut appuyer sur ou cliquez dessus pour le vérifier.
- Cases d'option sont normalement utilisés ensemble dans un RadioGroup.
- Lorsque plusieurs cases d'option sont à l'intérieur d'un groupe de boutons radio, la sélection d’une case d'option désactive toutes les autres.
- De même, vous pouvez appeler isChecked() sur un RadioButtonto voir si elle est activée, toggle() pour le sélectionner et ainsi de suite, comme vous pouvez le faire avec une case à cocher.

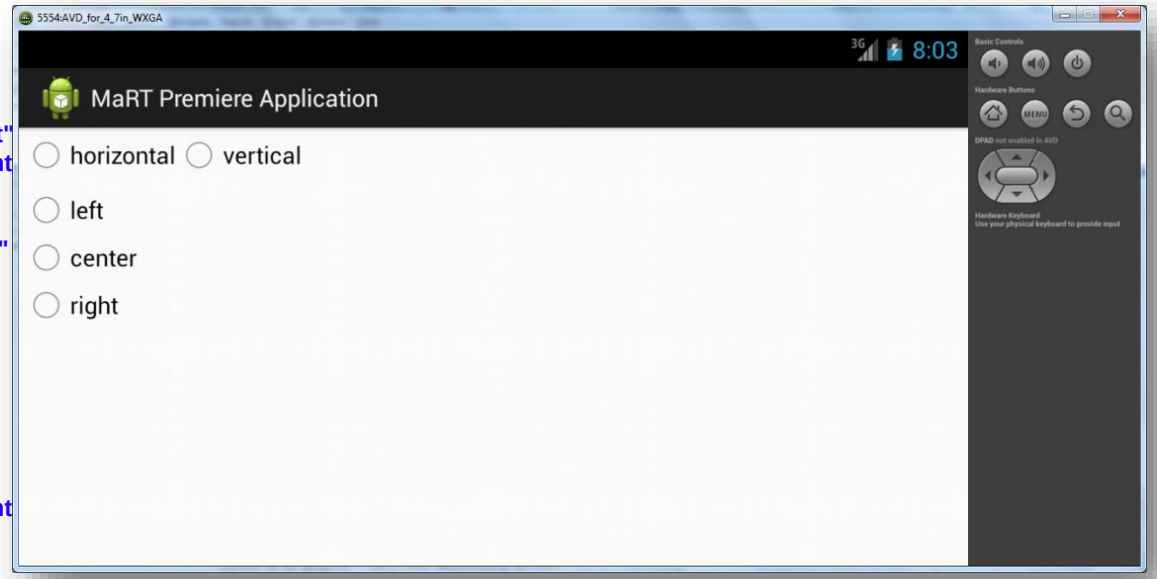
RadioGroup

- Pour grouper des boutons radio (ajoute un ascenseur si nécessaire)
- Un seul bouton peut être coché à la fois (attention à l'état initial qui n'est pris en compte par le RadioGroup que s'il est fait par la méthode **check du RadioGroup)**
- Exemple de fichier XML :

```
<RadioGroup
android:id="@+id/groupe_de_boutons"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:orientation="horizontal">
    <RadioButton
    …
    />
    <RadioButton
    …
    />
</RadioGroup>
```

Exemple RadioGroup

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <RadioGroup android:id="@+id/orientation"
                android:orientation="horizontal"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="5px">
                <RadioButton
                        android:id="@+id/horizontal"
                        android:text="horizontal" />
                <RadioButton
                        android:id="@+id/vertical"
                        android:text="vertical" />
        </RadioGroup>
        <RadioGroup android:id="@+id/gravity"
                android:orientation="vertical"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:padding="5px">
                <RadioButton
                        android:id="@+id/left"
                        android:text="left" />
                <RadioButton
                        android:id="@+id/center"
                        android:text="center" />
                <RadioButton
                        android:id="@+id/right"
                        android:text="right" />
        </RadioGroup>
</LinearLayout>
```



Basic Widgets: RadioButtons

Exemple

Cette interface utilisateu
utilise des cases d'optior
et cases à
cocher pour définir les
choix

Basic Widgets: RadioButtons

Nous étendons l'exemple précédent en ajoutant au RadioGroup trois RadioButton. Le nouveau code XML et Java seulement est produit :

activity_main.xml

```
58
59          <LinearLayout
60              android:id="@+id/monPanneau"
61              android:layout_width="132dp"
62              android:layout_height="wrap_content"
63              android:layout_margin="10dp"
64              android:background="#FFFF00"
65              android:orientation="vertical" >
66              <CheckBox
67                  android:id="@+id/dejeuner"
68                  android:layout_width="wrap_content"
69                  android:layout_height="wrap_content"
70                  android:layout_margin="10dp"
71                  android:text="@string/txtDejeuner" />
72              <CheckBox
73                  android:id="@+id/dessert"
74                  android:layout_width="wrap_content"
75                  android:layout_height="wrap_content"
76                  android:layout_margin="10dp"
77                  android:text="@string/txtDessert" />
78              <CheckBox
79                  android:id="@+id/the"
80                  android:layout_width="wrap_content"
81                  android:layout_height="wrap_content"
82                  android:layout_margin="10dp"
83                  android:text="@string/txtThe" />
84              <CheckBox
85                  android:id="@+id/boisson"
86                  android:layout_width="wrap_content"
87                  android:layout_height="wrap_content"
88                  android:layout_margin="10dp"
89                  android:text="@string/txtBoisson" />
90          </LinearLayout>
91
92      </LinearLayout>
```

Basic Widgets: RadioButtons

- Android Activity (1 of 3)

```
14  public class MainActivity extends Activity implements OnClickListener {
15      Button valider;
16      Button quitter;
17      CheckBox dejeuner;
18      CheckBox dessert;
19      CheckBox the;
20      CheckBox boisson;
21      RadioGroup cafeRadioGroup;
22      RadioButton expresso;
23      RadioButton capuccino;
24      RadioButton cafeTouba;
```

Basic Widgets: RadioButtons

- Android Activity (2 of 3)

```
26      @Override
27      protected void onCreate(Bundle savedInstanceState) {
28          super.onCreate(savedInstanceState);
29          setContentView(R.layout.activity_main);
30          valider = (Button) findViewById(R.id.button1);
31          quitter = (Button) findViewById(R.id.button2);
32          dejeuner = (CheckBox) findViewById(R.id.dejeuner);
33          dessert = (CheckBox) findViewById(R.id.dessert);
34          the = (CheckBox) findViewById(R.id.the);
35          boisson = (CheckBox) findViewById(R.id.boisson);
36
37          cafeRadioGroup = (RadioGroup) findViewById(R.id.radioGroup1);
38          expresso = (RadioButton) findViewById(R.id.radio0);
39          capuccino = (RadioButton) findViewById(R.id.radio1);
40          cafeTouba = (RadioButton) findViewById(R.id.radio2);
41
42          valider.setOnClickListener(this);
43          quitter.setOnClickListener(this);
44      }
```

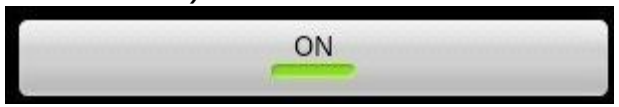
Basic Widgets: RadioButtons

MainActivity.java

```
53    @Override
54    public void onClick(View v) {
55        if (v.getId() == valider.getId()) {
56            String[] commandes = new String[5];
57            int i = 0;
58            if (dejeuner.isChecked())
59                commandes[i++] = "Déjeuner";
60            if (dessert.isChecked())
61                commandes[i++] = "Dessert";
62            if (the.isChecked())
63                commandes[i++] = "Thé";
64            if (boisson.isChecked())
65                commandes[i++] = "Boisson";
66            if (expresso.getId() == cafeRadioGroup.getCheckedRadioButtonId()) {
67                commandes[i++] = "Expresso";
68            }
69            if (capuccino.getId() == cafeRadioGroup.getCheckedRadioButtonId()) {
70                commandes[i++] = "Capucino";
71            }
72            if (cafeTouba.getId() == cafeRadioGroup.getCheckedRadioButtonId()) {
73                commandes[i++] = "Café Touba";
74            }
75
76            if (i == 0) {
77                Toast.makeText(getApplicationContext(),
78                        "Vous n'avez rien choisi !!!", Toast.LENGTH_LONG)
79                        .show();
80            } else {
81                String str = "Vous avez commandé:\n";
82                for (int j = 0; j < commandes.length; j++) {
83                    if (commandes[j] != null)
84                        str += "\t - " + commandes[j] + "\n";
85                }
86                Toast.makeText(getApplicationContext(), str, Toast.LENGTH_LONG)
87                        .show();
88            }
89        } else {
90            System.exit(0);
91        }
92    }
93
```

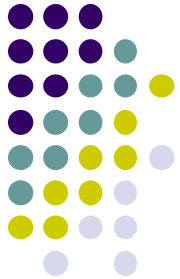
ToggleButton

- Mêmes paramètres XML que TextView plus :
  - android:disabledAlpha="x" (où est une valeur réelle entre 0 et 1) définit la transparence appliquée lorsque le bouton est inactif
  - android:textOff="txt" définit le texte quand le bouton n'est pas allumé
  - android:textOn="txt" définit le texte quand le bouton n'est pas allumé
  - android:checked="b" (où b vaut true ou false) place le bouton en position allumé ou éteint
- **Méthodes de la classe ToggleButton:** android.widget.ToggleButton
- ***Construction:*** ToggleButton(Context) le paramètre est généralement l'activité elle-même
- ***Etat***
  - isChecked() indique si le bouton est allumé ou éteint (true/false)
  - setChecked(boolean) positionne le bouton (true = allumé, false = éteint)
  - toggle() change l'état du bouton



- ***Aspect***
  - getTextOff() renvoie le texte associé au bouton quand il n'est pas allumé
  - getTextOn() renvoie le texte associé au bouton quand il est allumé
  - setTextOff(CharSequence) définit le texte associé au bouton quand il n'est pas allumé
  - setTextOn(CharSequence) définit le texte associé au bouton quand il est allumé

Autocomplétion : AutoCompleteTextView



- C’est une spécialisation de EditText pour apporter l’auto complétion
- Propriétés supplémentaires:

  android:completionHint="texte" texte affiché en titre du menu déroulant

  android:completionThreshold Pour définir le nombre de caractères à taper avant que la complétion n'entre en action.

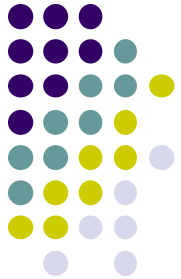
  android:dropDownHeight="unité" on peut aussi utiliser les constantes fill_parent et wrap_content, définit la hauteur du menu déroulant

  android:dropDownWidth="unité" on peut aussi utiliser les constantes fill_parent et wrap_content, définit la hauteur du menu déroulant

  android:dropDownHorizontalOffset Pour définir le décalage horizontal du menu déroulant

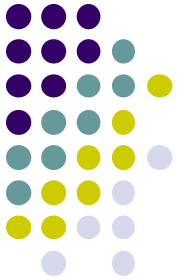
  android:dropDownVerticalOffset Pour définir le décalage vertical du menu déroulant

Autocomplétion : AutoCompleteTextView



- ***Construction***
  - AutoCompleteTextView(Context) le paramètre est généralement l'activité elle-même
- ***Aspect et contenu***
  - showDropDown() montre la liste déroulante
  - dismissDropDown()cache la liste déroulante
  - getListSelection() renvoie le rang de la proposition choisie
  - setListSelection(int) choisit une proposition par son rang
  - setAdapter(ArrayAdapter) Permet de remplir la liste de propositions. La classe ArrayAdapter est décrite dans ListView.
  - setCompletionHint(CharSequence) définit le texte affiché en titre du menu déroulant

Autocomplétion: exemple

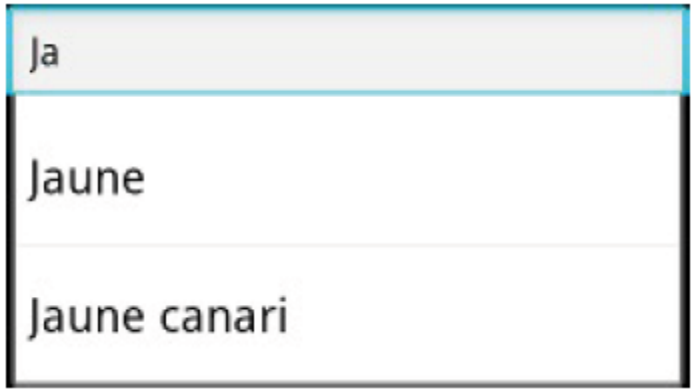


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="vertical" >
      <AutoCompleteTextView
            android:id="@+id/complete"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
</LinearLayout>
```

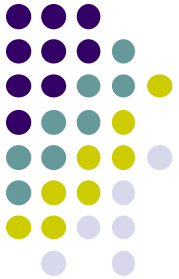
Autocomplétion: exemple

- Déclarer l'activité AutoCompletionActivity suivante :

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
public class AutoCompletionActivity extends Activity {
        private AutoCompleteTextView complete = null;
        // Notre liste de mots que connaîtra l'AutoCompleteTextView
        private static final String[] COULEUR = new String[] {
                "Bleu", "Vert", "Jaune", "Jaune canari", "Rouge", "Orange"
        };
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                // On récupère l'AutoCompleteTextView déclaré dans notre layout
                complete = (AutoCompleteTextView) findViewById(R.id.complete);
                complete.setThreshold(2);
                // On associe un adaptateur à notre liste de couleurs…
                ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line, COULEUR);
                // … puis on indique que notre AutoCompleteTextView utilise cet adaptateur
                complete.setAdapter(adapter);
        }
}
```



ImageView : Afficher des images



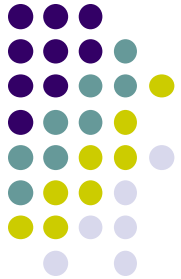
- Permet d'afficher des images
- Propriétés :
  - Contenu
    - android:src Pour définir une couleur ou une image.
    - android:tint Pour définir une couleur qui teinte l'image
  - Position et dimensions de l'image
    - android:adjustViewBounds La taille de l'ImageView sera ou pas modifiée
    - android:baselineAlignBottom Cadrage ou pas de l'image en bas de la zone
    - android:cropToPadding L'image sera ou pas coupée si elle est plus grande que la taille disponible
    - android:scaleType Pour définir le mode de redimensionnement de l'image avec ou sans déformation. (voir exemples transparent suivant)
  - Taille
    - android:maxHeight Pour définir la hauteur maximale
    - android:maxWidth Pour définir la largeur maximale

ImageView : Afficher des images

```
<ImageView android:id="@+id/image"
    android:src="@drawable/keithwembley"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:maxHeight="200px"
    android:adjustViewBounds="true"
    android:scaleType="centerCrop"/>
```

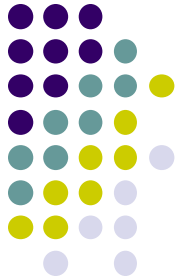
```
<ImageView android:id="@+id/image"
    android:src="@drawable/keithwembley"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:maxHeight="200px"
    android:adjustViewBounds="true"
    android:scaleType="fitXY"/>
```

La gestion des evenements



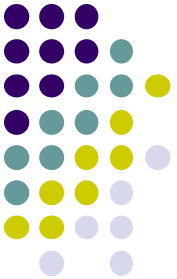
- la programmation des applications devient intéressant quand on peut capturer les actions de l'utilisateur pour interagir avec lui.
- Sous Android, les Listener permettent de spécifier le comportement des éléments en fonction des évènements que l'utilisateur génère. Les principaux Listeners sont: **onClickListener, onLongClickListener , onTouchListener, onCreateContextMenuListener, onFocusChangeListener, onKeyListener…**

La gestion des evenements



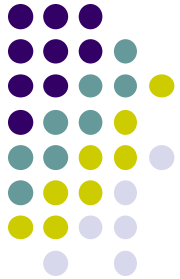
- **onClickListener** - Utilisé pour détecter des événements click qui correspond à un appui suivi d'un relachement d'une zone de l'ecran qui contient une vue. Ce Listener permet créer un CallBack sur l'evenement onClick().
- **onLongClickListener** - Utilisé pour détecter lorsque l'utilisateur maintient le contact sur une vue pendant une période prolongée. Correspond à la méthode de rappel onLongClick () qui est passé à la vue qui a reçu l'événement comme un argument.

La gestion des evenements



- **onTouchListener** - utilisés pour détecter toute forme de contact avec l'écran tactile, y compris des touches individuelles ou multiples et des mouvements gestuels. Correspondant à la fonction OnTouch().
- **onCreateContextMenuListener** - Attend la création d'un menu contextuel comme le résultat d'un long clic. Correspond à la méthode de rappel onCreateContextMenu ().

La gestion des evenements



- **onFocusChangeListener** - Détecte lorsque le focus se déplace vers une autre vue à la suite d'une interaction avec un track-ball ou la touche de navigation. Correspond à la onFocusChange ().
- **onKeyListener** - Utilisé pour détecter quand une touche sur le clavier est enfoncé. Correspond à la méthode OnKey ().

La gestion des evenements

- Un exemple de gestion de l'evenement est illustré par ce bouton.

```
Button button = (Button)findViewById(R.id.myButton);

button.setOnClickListener(
        new Button.OnClickListener() {
           public void onClick(View v) {
              TextView myTextView =
                    (TextView)findViewById(R.id.myTextView);
              myTextView.setText("A click on the button");
           }
        }
);

button.setOnLongClickListener(
        new Button.OnLongClickListener() {
           public boolean onLongClick(View v) {
              TextView myTextView =
                    (TextView)findViewById(R.id.myTextView);
              myTextView.setText("A long click on the button");
              return true;
           }
        }
);
```

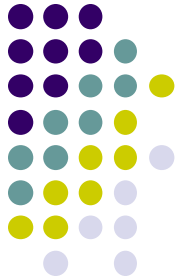
Evaluation

- **Exercice 1:** Browse documentation concerning TextView widget at http://developper.android.com/reference/android/widget/TextView and fill in the following table:

Property	Description
android:autoLink	
android:capitelize	
android:editable	
android:fontFamily	
android:gravity	
android:text	
android:textSize	
android:width	
android:height	

- Find in the same page, methods that allow retrieving and modifing value of the attributes in java program.

Evaluation



- **Exercice 2:** Write an application “change my background color” that contains an EditText and a Button. Each click on the button will change randomly the background color of EditText.

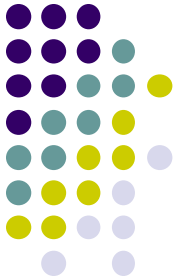
Développement d'applications mobiles

ANDROID - GROUPS

M2 RT/GLAR

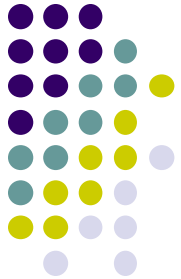
PARTIE 5

Pr. El hadji M. NGUER



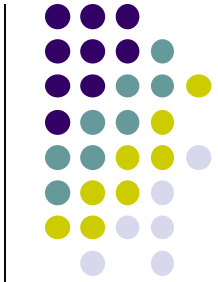
CONTAINERS

Groups



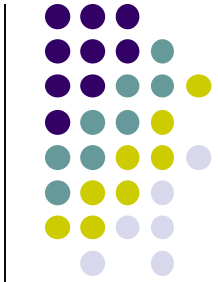
- Group items participant a choice:
  - **ListView** (several elements organized in a vertical list with dividers). Often used for word lists (menu type).
  - **GridView** (a number of items in table). Often used for word tables (menu type).
  - **RadioGroup** (radio group which only one can be checked at once) - See previous section
  - **Gallery** (more elements arranged horizontally scrolling). Often used for images

ListView



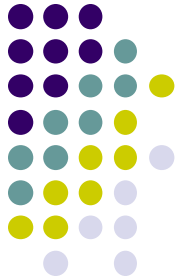
- Place the vertical list elements and adds a lift if necessary
  - ***Separators***
    - android:divider For setting the color of the separators or use an image as a separator.
    - android:dividerHeight="unity" In order to define the height of the separators (even if they contain an image)
  - ***Type of choice***
    - android:choiceMode="c" (où c can take values : none, singlechoice, multipleChoice) to indicate method of choice in the list (none, one, several).

ListView (Content)



- ***In XML (text only)***
  - android: entries = "@ array / mylist" defines the contents of the list from the contents of an XML file placed in res / values /
- ***Using a content management system (Adapter)***
  - setAdapter(Adapter) to associate to ListView
  - ***Either predefined class (ArrayAdapter , SimpleAdapter, CursorAdapter)***
    - ArrayAdapter(Context, type) second parameter is of predefined type :
      - android.R.layout.simple_list_item_1 for a unique choice list or
      - android.R.layout.simple_list_item_multiple_choice for a multiple choice list (a checkbox appears next to each item in the list)
    - ArrayAdapter.add(Object) pour remplir la liste
  - ***Or custom class (inheritance Base Adapter)***

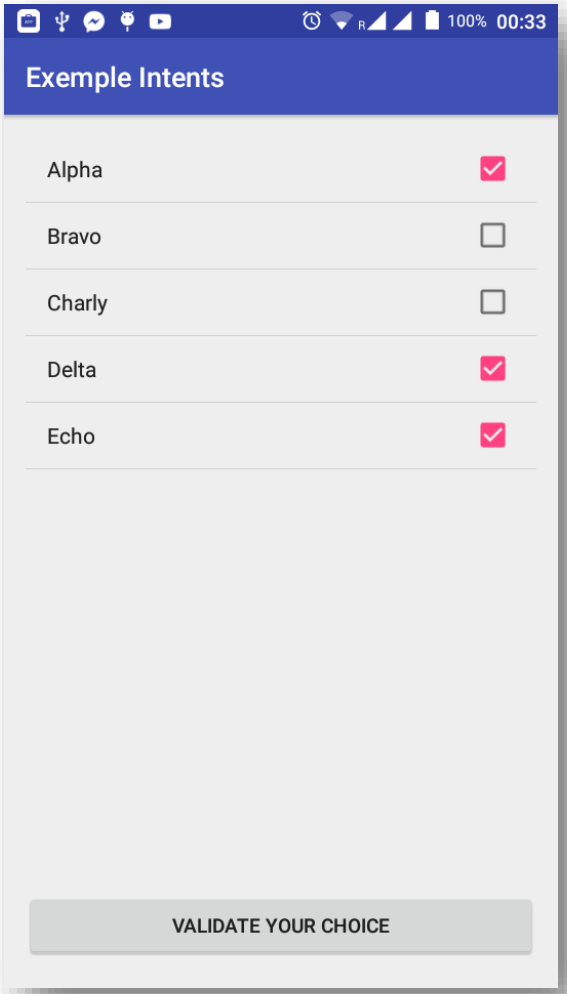
Precision about adapters

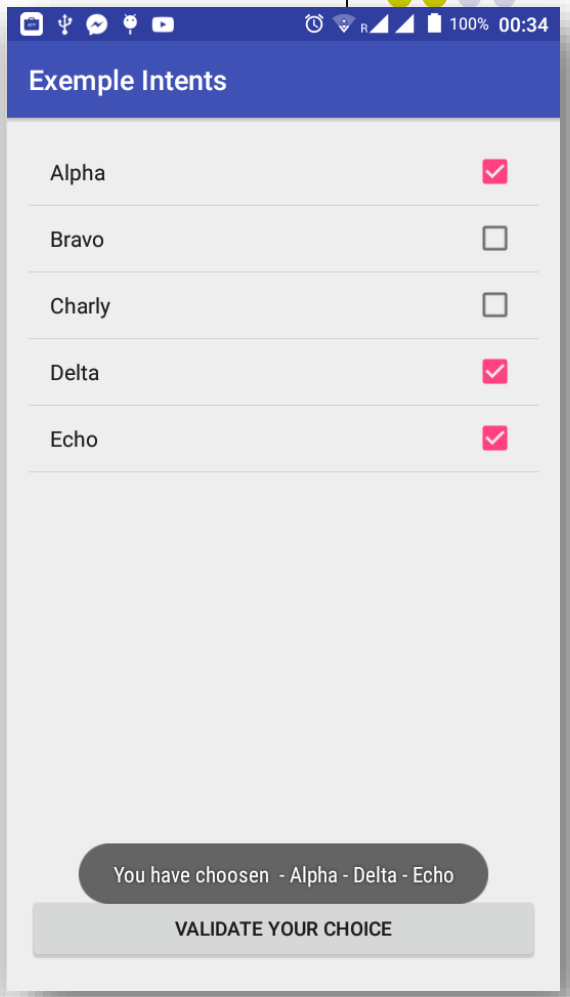


- The adapters are classes that bind data to the views of the UI
  - Concerned views extends android.widget.AdapterView : Spinner, Gallery, GridView, ListView that are subclass of AdapterView
- Adapters classes
  - Subclass of android.widget.BaseAdapter
  - SimpleAdapter, ArrayAdapter<?> : is used to retrieve data stored in a collection
- Exploits by defaults toString() Method of objects of the list
  - CursorAdapter : used to retrieve data stored in a relational database (SQLite)
  - You can also extend these base classes to finely handle your items (recommended)

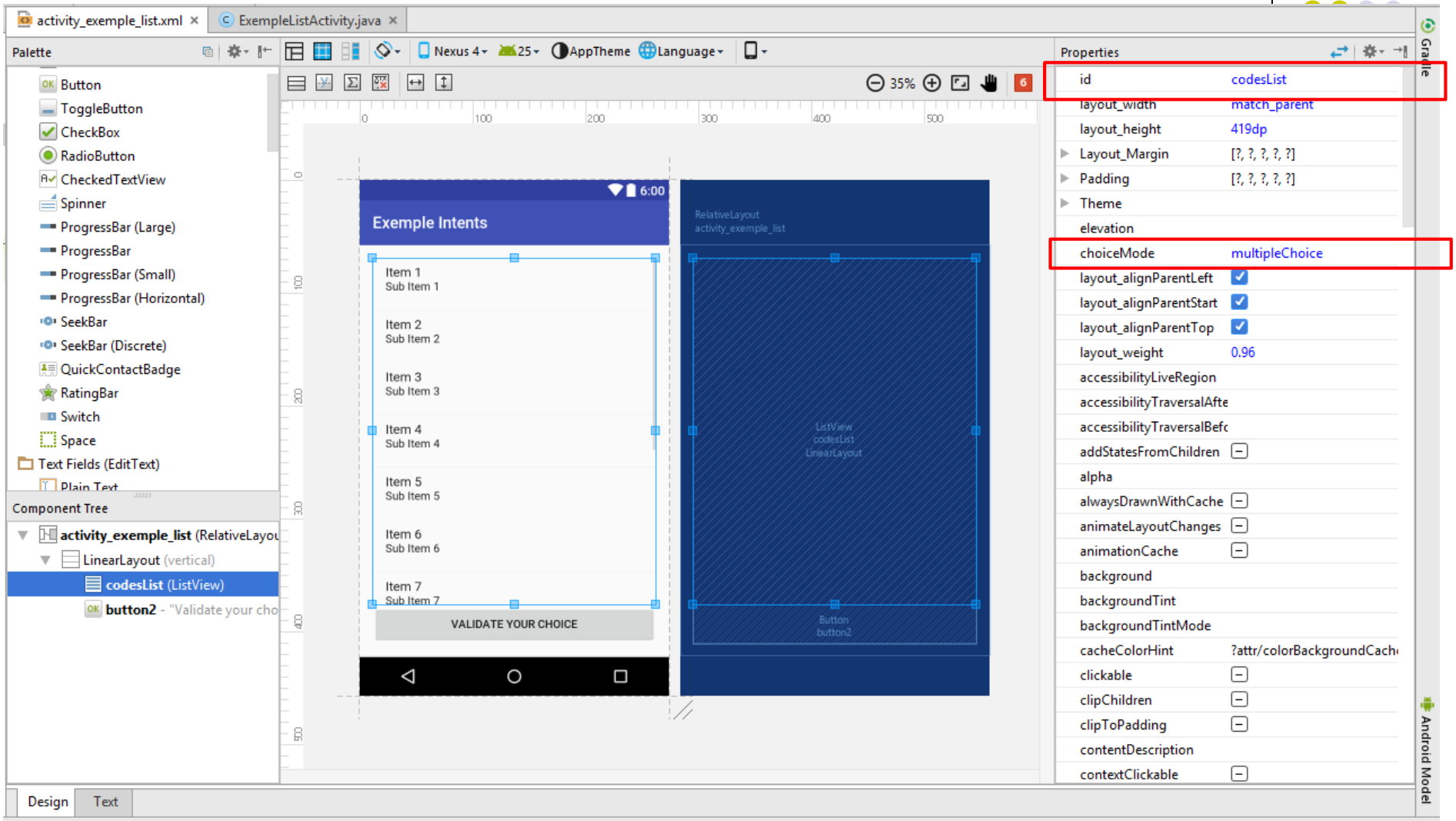
Example of ListView

1. Create a new empty activity
2. Place a ListView container on its layout file
3. Add button for validation

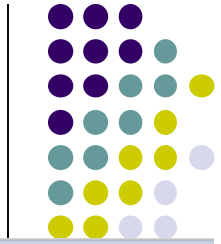




List layout Properties



List layout Properties



ExempleListActivity.java × activity_exemple_list.xml ×

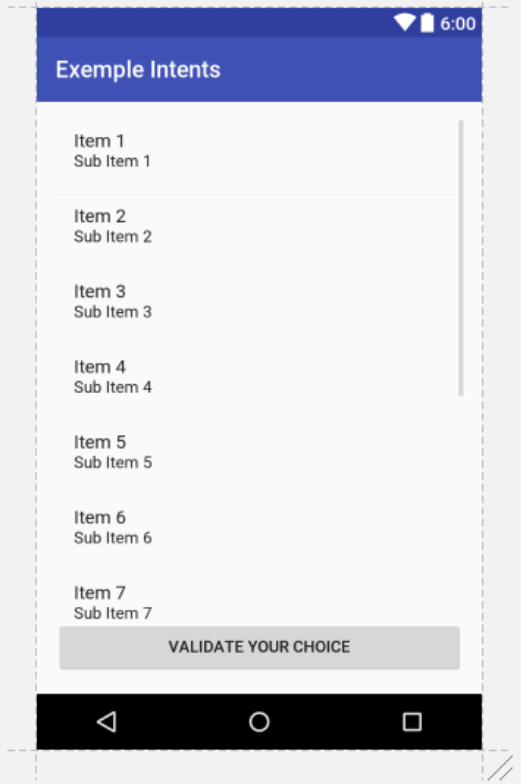
RelativeLayout | LinearLayout | Button

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_exemple_list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.toh.exempleintents.ExempleListActivity">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:weightSum="1">

        <ListView
            android:layout_width="match_parent"
            android:layout_height="419dp"
            android:layout_alignParentTop="true"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:layout_weight="0.96"
            android:id="@+id/codesList"
            android:choiceMode="multipleChoice" />

        <Button
            android:text="Validate your choice"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/button2"
            android:onClick="validateChoice (ExempleListActivity)" />
    </LinearLayout>

</RelativeLayout>
```

Design | Text

Preview

Nexus 4 | 25 | AppTheme | Language | 39%

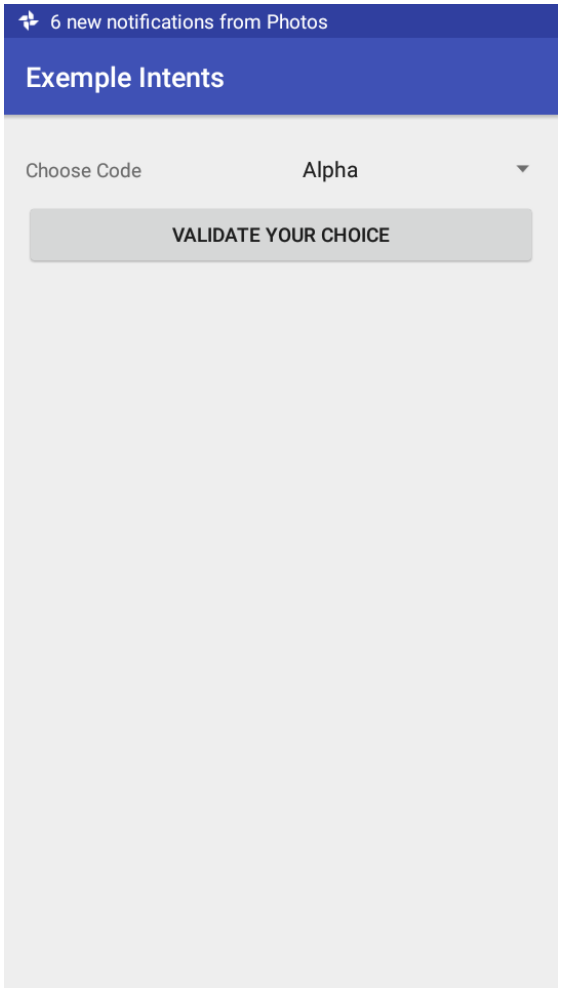


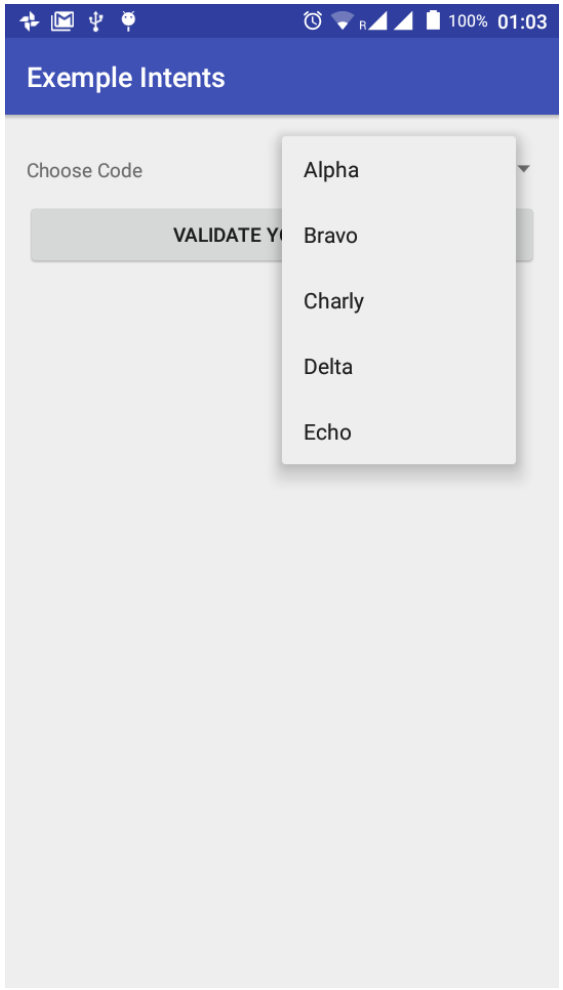
Activity: java side

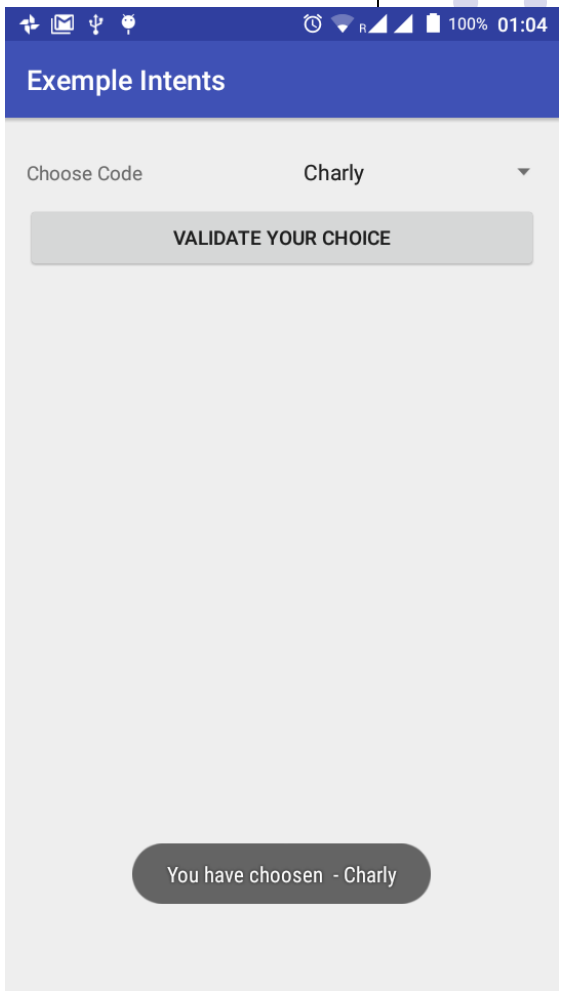
ExempleListActivity.java × activity_exemple_list.xml ×

```
1   package com.example.toh.exempleintents;
2
3   import ...
11
12  public class ExempleListActivity extends AppCompatActivity {
13      ListView codesListView;
14      String[] myStringArray;
15
16      @Override
17      protected void onCreate(Bundle savedInstanceState) {
18          super.onCreate(savedInstanceState);
19          setContentView(R.layout.activity_exemple_list);
20          // 1. Create a string array to hold the text to be displayed
21          myStringArray = new String[]{"Alpha", "Bravo", "Charly", "Delta", "Echo"};
22
23          //2. Create the ArrayAdapter
24          ArrayAdapter myArrayAdapter = new ArrayAdapter(this, android.R.layout.simple_list_item_multiple_choice, myStringArray);
25
26          //3. Associate the ArrayAdapter with the ListView
27          codesListView = (ListView) findViewById(R.id.codesList);
28          codesListView.setAdapter(myArrayAdapter);
29      }
30
31      public void validateChoice(View v) {
32          SparseBooleanArray pos = codesListView.getCheckedItemPositions();
33          String choix = "";
34          for (int i = 0; i < pos.size(); i++) {
35              choix += " - " + myStringArray[pos.keyAt(i)];
36          }
37
38          Toast.makeText(getApplicationContext(), "You have choosen " + choix, Toast.LENGTH_LONG).show();
39      }
40  }
```

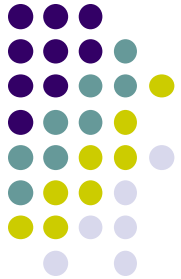
Choice List (Spinner)





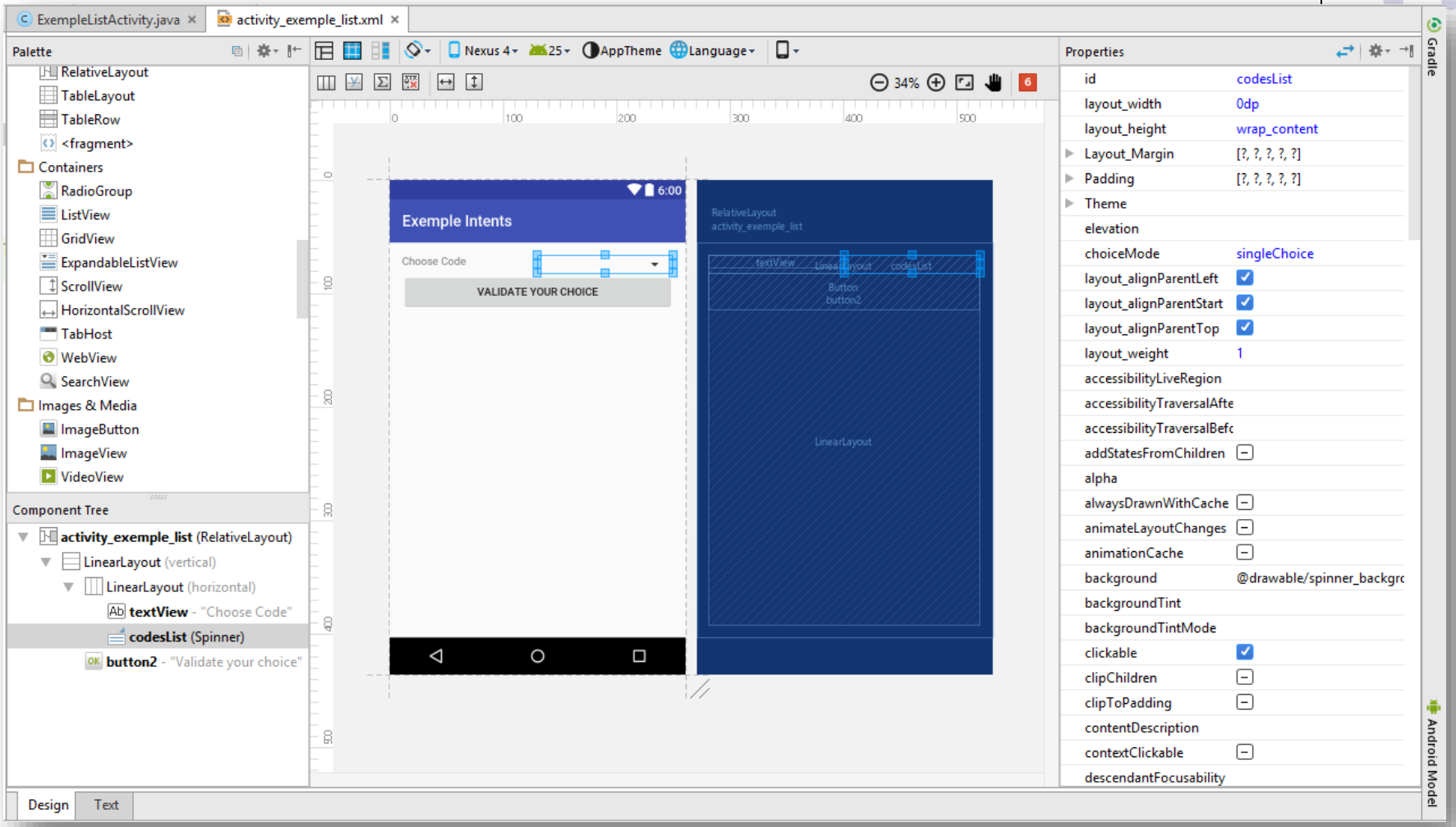


Choice List (Spinner)

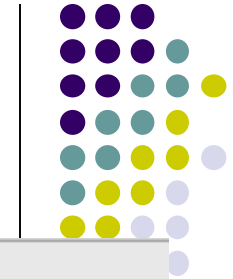


- Displays current selection and displays a RadioGroup when clicked to change
- properties:
  - android: prompt To set the title of the window that opens when making a choice
  - android: entries = "@ array / mylist" defines the contents of the list from the contents of the file string.xml placed in res / values / which has the form of the file in front:

Choice List (Spinner)



Choice List (Spinner)



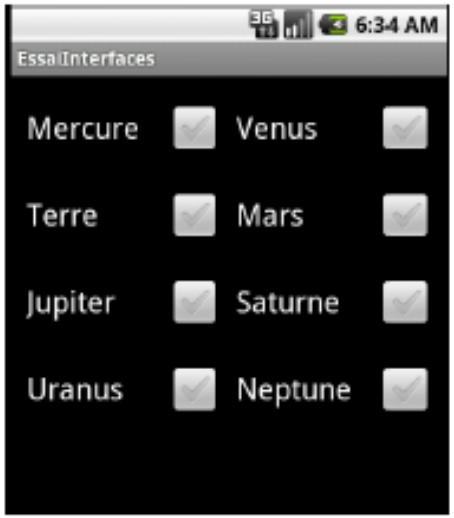
ExempleListActivity.java × activity_exemple_list.xml ×

```
1  package com.example.toh.exempleintents;
2
3  import ...
9
10 public class ExempleListActivity extends AppCompatActivity {
11     Spinner codesSpinnerView;
12     String[] myStringArray;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_exemple_list);
18         // 1. Create a string array to hold the text to be displayed
19         myStringArray = new String[]{"Alpha", "Bravo", "Charly", "Delta", "Echo"};
20
21         //2. Create the ArrayAdapter
22         ArrayAdapter myArrayAdapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1, myStringArray);
23
24         //3. Associate the ArrayAdapter with the ListView
25         codesSpinnerView = (Spinner) findViewById(R.id.codesList);
26         codesSpinnerView.setAdapter(myArrayAdapter);
27     }
28
29     public void validateChoice(View v) {
30         String choix= " - " + myStringArray[codesSpinnerView.getSelectedItemPosition()];
31         Toast.makeText(getApplicationContext(), "You have choosen " + choix, Toast.LENGTH_LONG).show();
32     }
33 }
```

GridView

- Fonctionne comme ListView mais permet une présentation en plusieurs colonnes
- Exemple
  - Dans le XML d'interface

```
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/liste_de_planetes"
  android:entries ="@array/planetes"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:numColumns="2"
  android:stretchMode="columnWidth"
  android:columnWidth="60dp"
  android:gravity="fill_horizontal"
  android:choiceMode="multipleChoice"
  />
```



  - Dans le code (même principe)

```
GridView table = (GridView) findViewById(R.id.liste_de_planetes);
String[] elements = getResources().getStringArray(R.array.planetes);
ArrayAdapter<String> adaptateur = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_multiple_choice);
for (int i=0; i<elements.length; i++) adaptateur.add(elements[i]);
table.setAdapter(adaptateur);
```

Gallery

- Normalement utilisé pour faire des galeries d'images avec défilement horizontal
- Propriétés supplémentaires
  - android:animationDuration Pour définir la durée de la transition (en ms) lorsque l'on passe d'un élément à l'autre
  - android:unselectedAlpha Pour définir la transparence des éléments non sélectionnés
- Pour remplir une galerie il faut un Adapter (comme pour ListView) mais que l'on doit écrire par héritage de la classe BaseAdapter puis l'associer à la galerie par la méthode setAdapter

Exemple de fichier XML :

```
<Gallery android:id="@+id/magalerie"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:unselectedAlpha="0.5"
/>
```

Développement d'applications mobiles

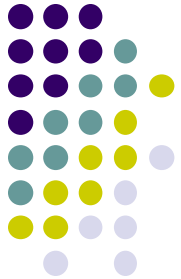
Android Fonts

M2 RT/GLAR

PARTIE 6

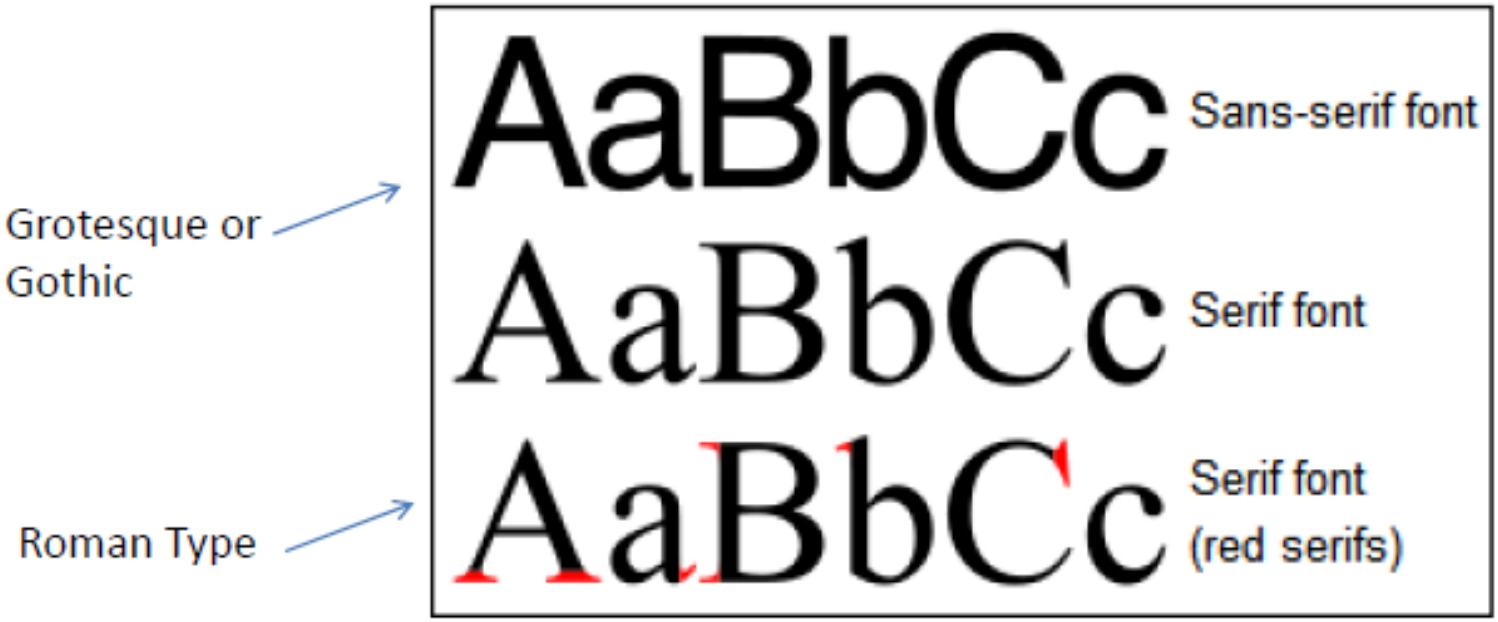
Pr. El hadji M. NGUER

Fonts



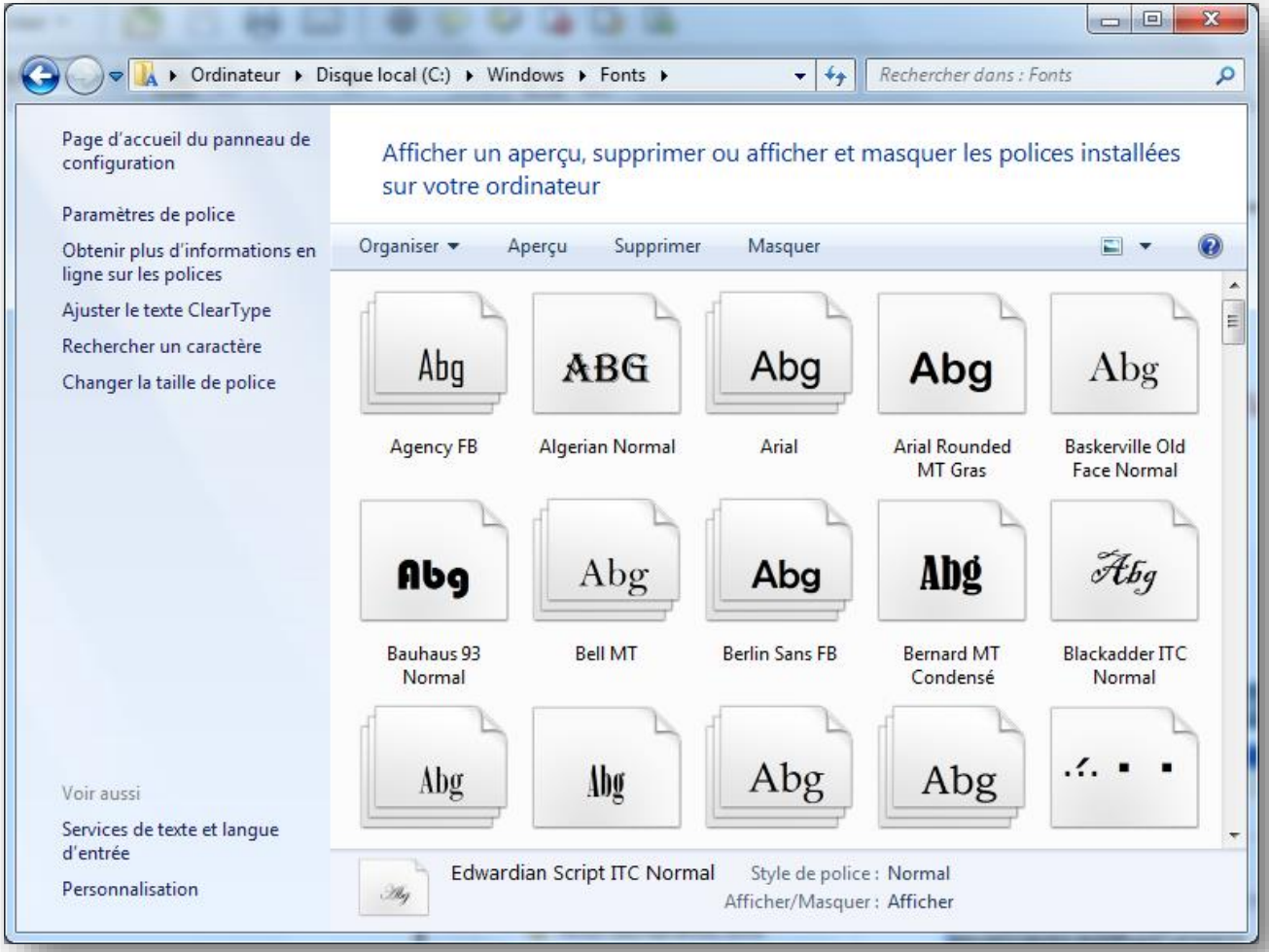
- Android propose *naturaly* three fonts family :
  - **sans**: Une étudiante fantôme en CFPP
  - **serif**: Une étudiante fantôme en CFPP
  - **monospaced**: `Une étudiante fantôme en CFPP`

Fonts

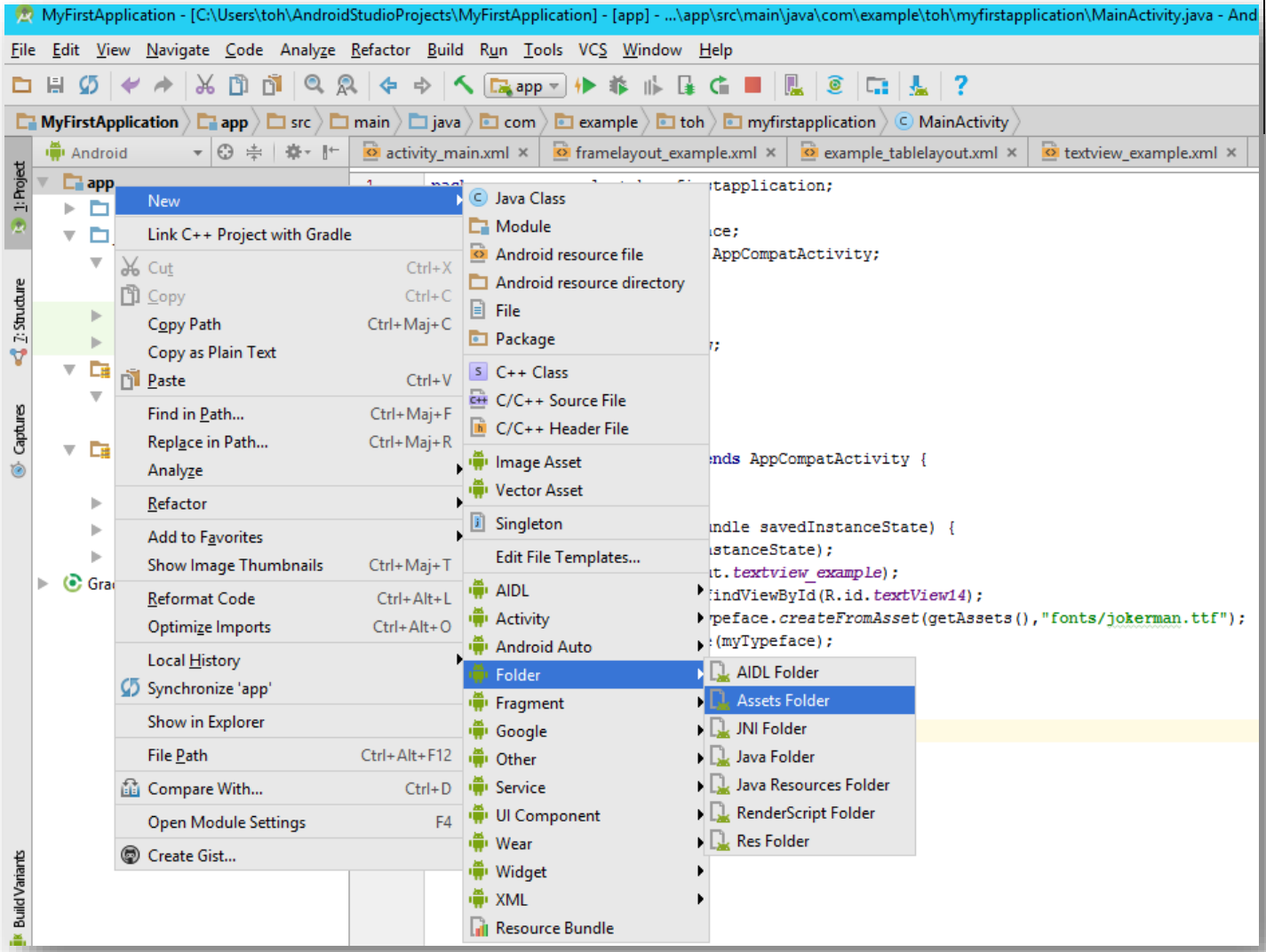


Source: http://en.wikipedia.org/wiki/Serif

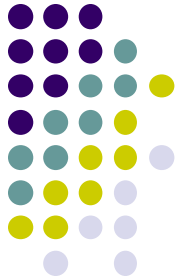
Other Fonts(Windows)



Fonts: 1. Create Assets folder



Fonts: Second step

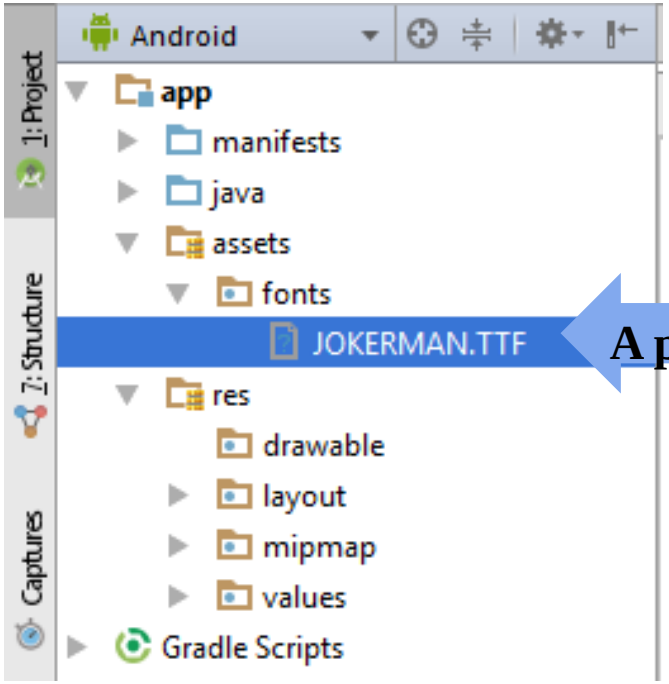


- To add a new family to make your application:

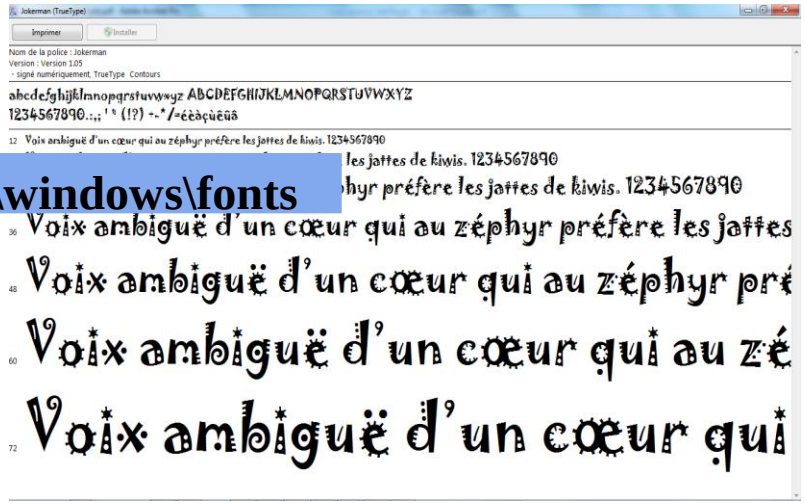
1. Create a directory / fonts in / assets
2. Copy or fonts you want to use this directory.
3. Use java to apply the font (see example)

Fonts

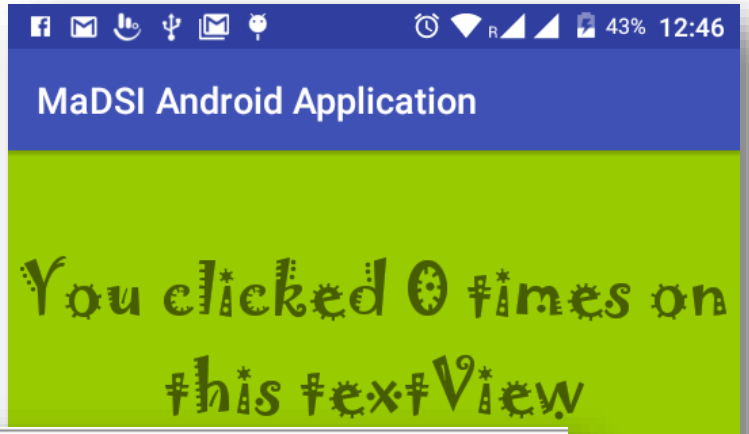
- **Example**: Using the Jokerman.TTF font.



A partir de c:\windows\fonts



Fonts




```
1  package com.example.toh.myfirstapplication;
2
3  import android.graphics.Typeface;
4  import android.support.v7.app.AppCompatActivity;
5  import android.os.Bundle;
6  import android.view.View;
7  import android.widget.Button;
8  import android.widget.TextView;
9  import android.widget.Toast;
10
11 import org.w3c.dom.Text;
12
13 public class MainActivity extends AppCompatActivity {
14     TextView myTextView;
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.textview_example);
19         myTextView=(TextView)findViewById(R.id.textView14);
20         Typeface myTypeface=Typeface.createFromAsset(getAssets(),"fonts/jokerman.ttf");
21         myTextView.setTypeface(myTypeface);
22     }
23 }
```

Développement d'applications mobiles

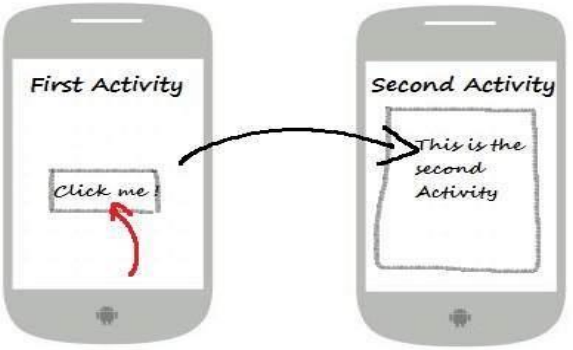
ANDROID- INTENTS

M2 RT/GLAR

PARTIE 7

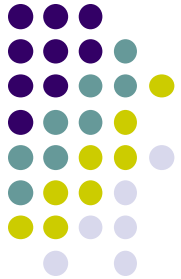
Pr. El hadji M. NGUER

Intents launch Activities



- Each screen with a user interface is an Activity
  - A note writing screen, a list of notes, app settings, …
  - The user is focused on doing one thing
- A component in your app implements one Activity
- Your app usually has multiple Activities (so, also multiple components)
- Components invoke other Activities using Intents
  - Simply pass an **Intent** object to **startActivity()**
  - Activities in other apps can also be invoked – for example, viewing web pages can invoke Activity in the browser app
  - Your app can publish any **Activity** it implements via its manifest file

Intents for a new activity



- Many ways to create the type of Intent object that will start a new activity. If one goes out to an internal activity to the application, you can create and pass the Intent class activity targeted by Intent:

```
Intent login = new Intent(this, SeLoguer.class);
startActivity(login);
```

Intents for a new activity

If it's to hand over to another application, it gives the manufacturer the Intent of the data and the target URI: the OS is responsible for finding an application that can meet the Intent.

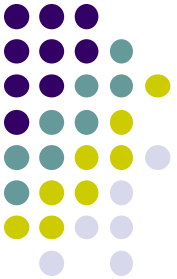
```
myButton = (Button) findViewById(R.id.button2);
myButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Uri telnumber = Uri.parse("tel:00221772051779");
        Intent call = new Intent(Intent.ACTION_CALL, telnumber);
        if (ActivityCompat.checkSelfPermission(getApplicationContext(), Manifest.permission.CALL_PHONE) == PackageManager.PERMISSION_GRANTED) {
            startActivity(call);
        }
    }
});
```

Do not forget the permission in the Manifest

manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.toh.myfirstapplication">

    <uses-permission android:name="android.permission.CALL_PHONE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MaDSI Android Application"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Intent with results expected in return

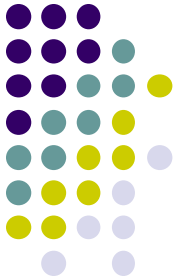


- When the back button is pressed, the current activity ends and returns to the previous activity. This allows for example to finish his phone call and return to the one who initiated the call interface.
- Within an application, a business may want to recover the activity of return code "child". This is done using the method **startActivityForResult** that sends a return code to the child activity.

  public void startActivityForResult (Intent intent, int requestCode)
- When the relative activity takes over, it becomes possible to filter the return code in the method **onActivityResult** whether or not we return the child activity.

```
protected void onActivityResult(int requestCode,
                                int resultCode,
                                Intent data)
```

Return from an activity



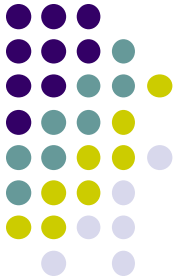
- The call of a Intent becomes in the parent activity:

```
public void onCreate(Bundle savedInstanceState) {
        Intent login = new Intent(getApplicationContext(),
        GetPhoneNumber.class);
        startActivityForResult(login,48);
        ...
 }
```

- Filtering in the parent class to see who had called this child activity:

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == 48)
                Toast.makeText(this,
                        "Request code recover (I know from where I come)",
                        Toast.LENGTH_LONG).show();
                }
}
```

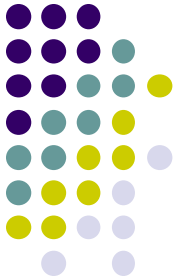
Result of an activity



- It is also possible to define the result of activity, to explicitly call before the end from an activity with the method **finish()**. In this case, the **setResult** method lets you record a return code that will also be possible to filter in the parent activity.
- In the child activity, therefore puts :

```
Button terminate= (Button)findViewById(R.id.terminate);
terminate.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
                setResult(RESULT_OK, getIntent());
                finish();
        }
});
```

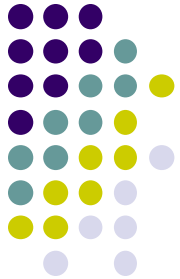
Result of an activity



- And the parent class can be filtered as follows:

```
protected void onActivityResult(int requestCode, int resultCode, Intent data){
        if (requestCode == 48)
                Toast.makeText(this, "Request code recover (I know from where I come)",
                                Toast.LENGTH_LONG).show();
        if (requestCode == 50)
                Toast.makeText(this, "Return code ok (I was sent the correct code)",
                                Toast.LENGTH_LONG).show();
}
```

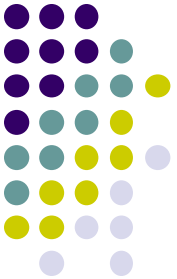
Adding Information



- The ***Intent*** possible to carry information to the target activity. We call these information **Extra**: methods for manipulating them are **getExtra** and **putExtra**.
- When preparing an Intent and you want to add a type of information "Key -> value" we do this:

```
Intent callactivity2 = new Intent(getApplicationContext(), Activity2.class);
callactivity2.putExtra("login", "Lamane");
startActivity(callactivity2);
```

Adding Information



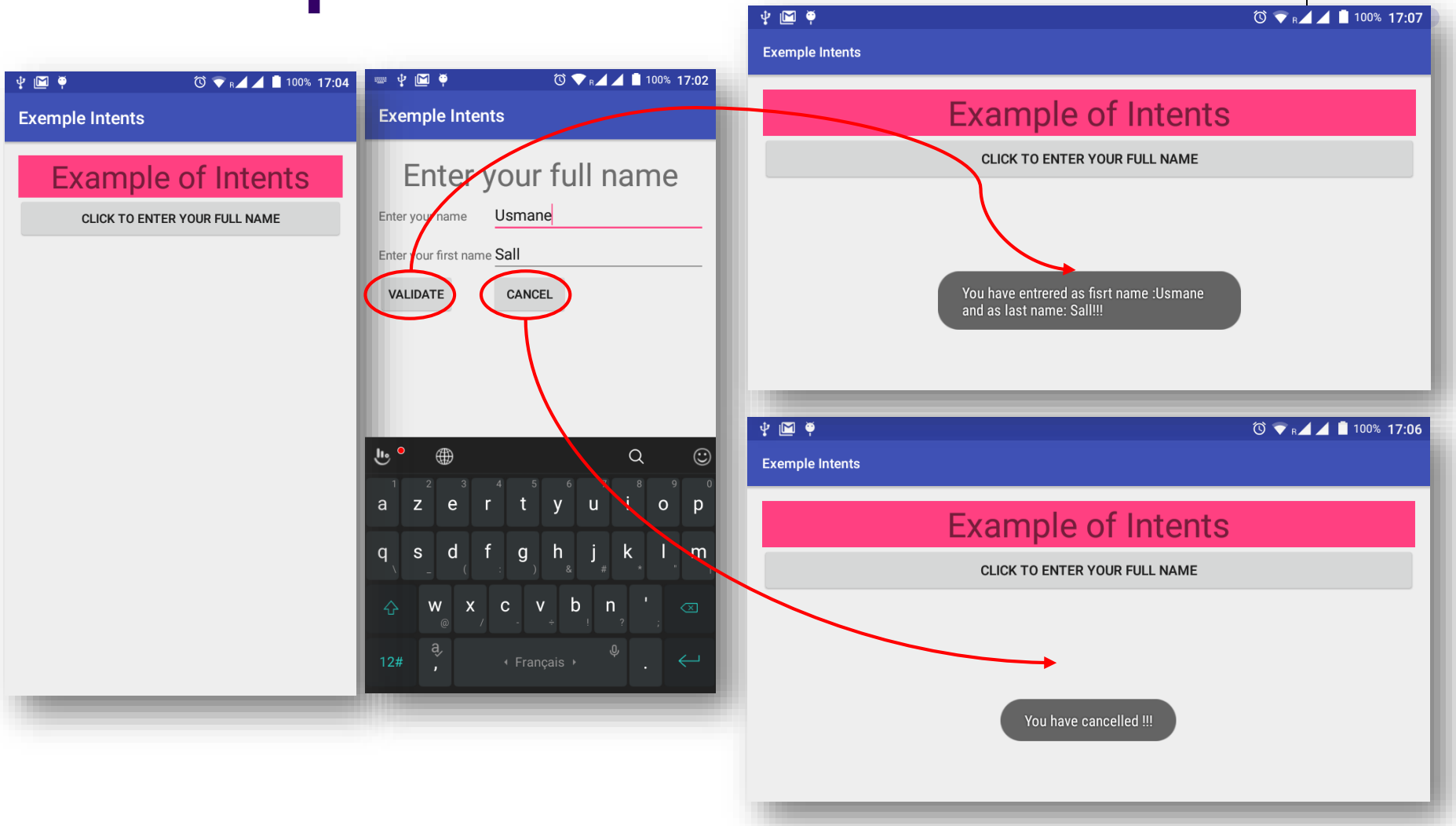
- On the side of the activity receiving Intent information is recovered as follows:

```
Bundle extras = getIntent().getExtras();
String s = new String(extras.getString("login"));
```

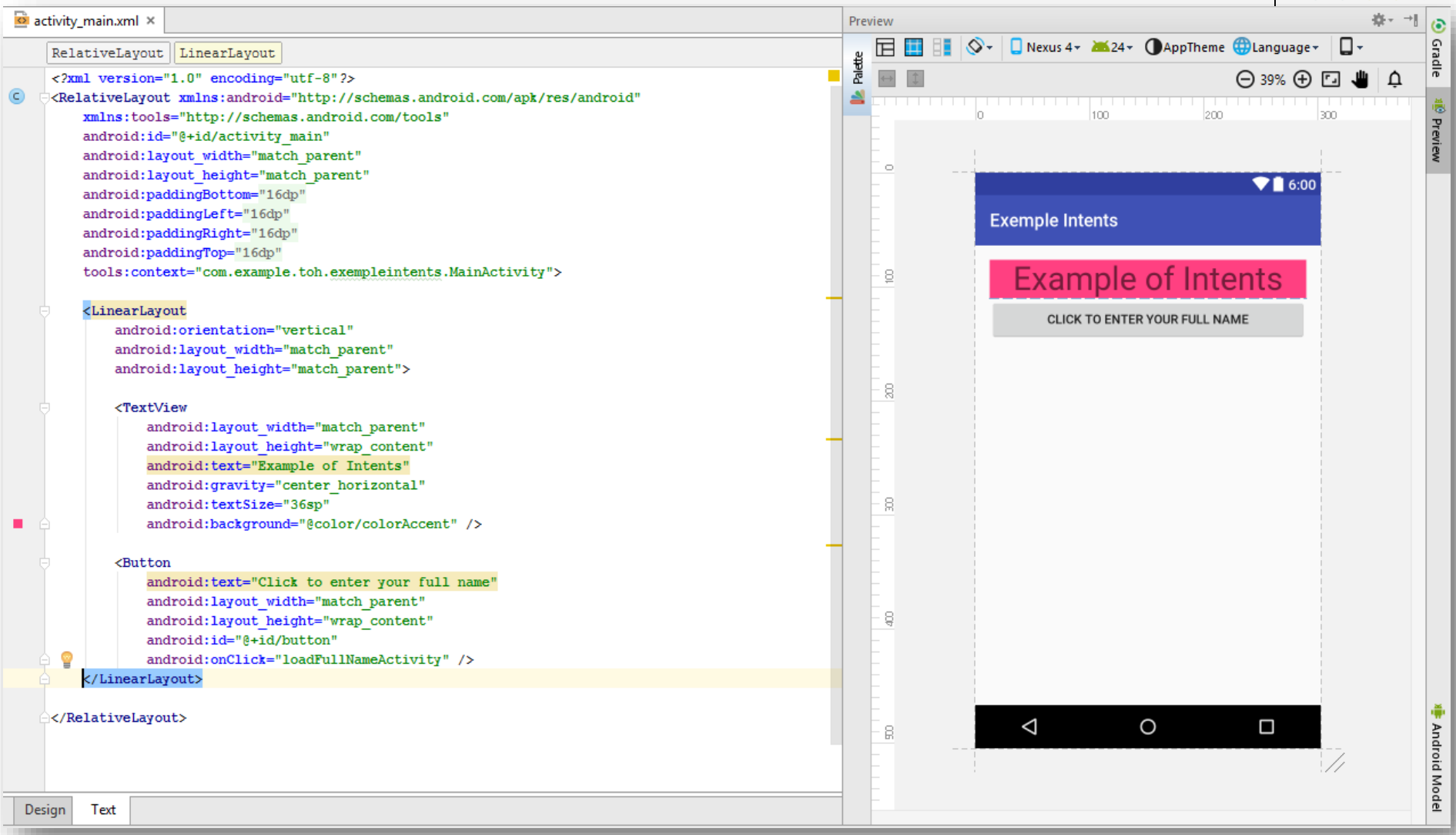

100% 17:04
Exemple Intents
Example of Intents
CLICK TO ENTER YOUR FULL NAME

Example 1

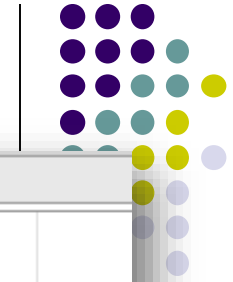
Example 1


100% 17:04
Exemple Intents
Example of Intents
CLICK TO ENTER YOUR FULL NAME
100% 17:02
Exemple Intents
Enter your full name
Enter your name
Usmane
Enter your first name
Sall
VALIDATE
CANCEL
100% 17:07
Exemple Intents
Example of Intents
CLICK TO ENTER YOUR FULL NAME
You have entrered as fisrt name :Usmane
and as last name: Sall!!!
100% 17:06
Exemple Intents
Example of Intents
CLICK TO ENTER YOUR FULL NAME
You have cancelled !!!

Main Activity



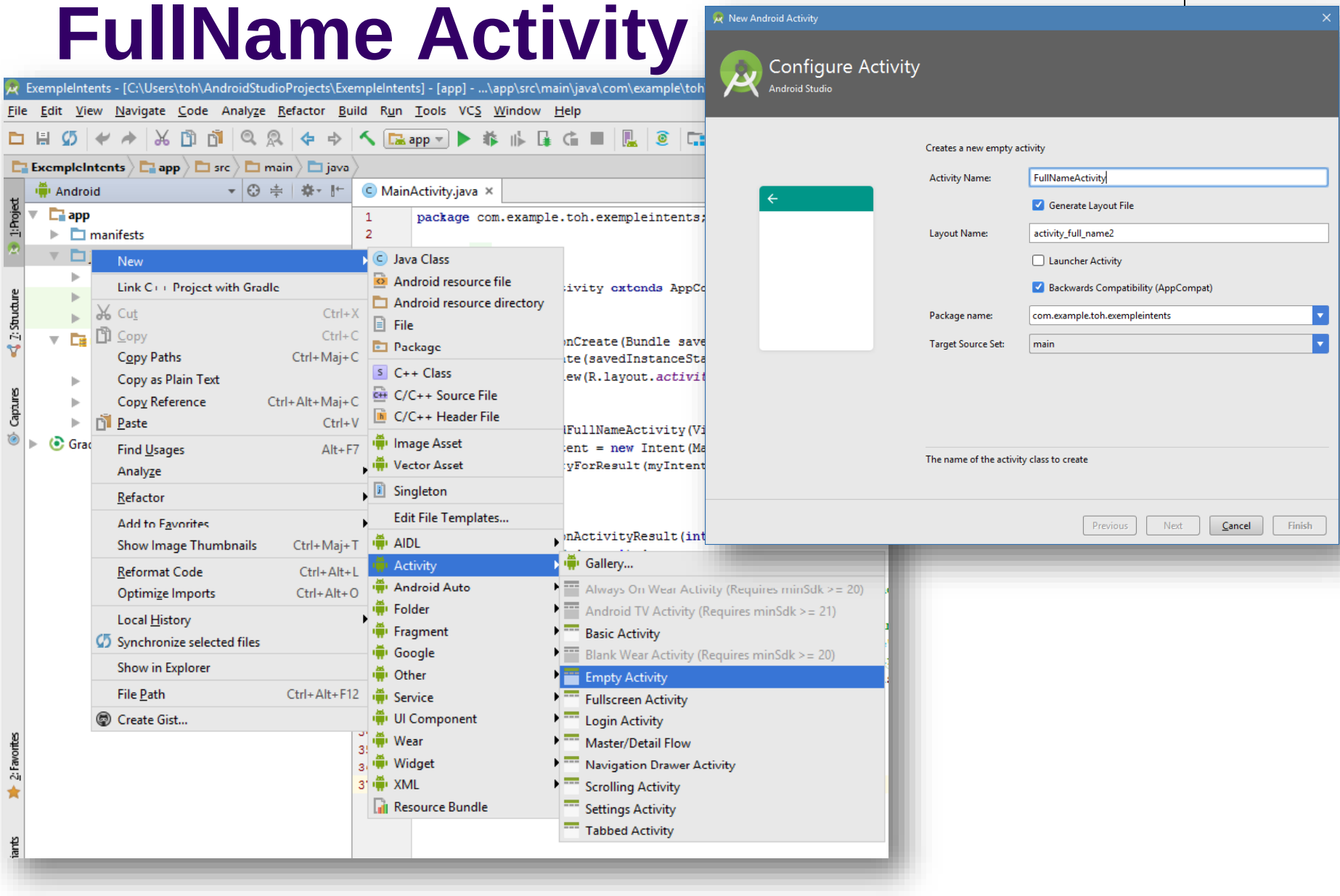
Main Activity



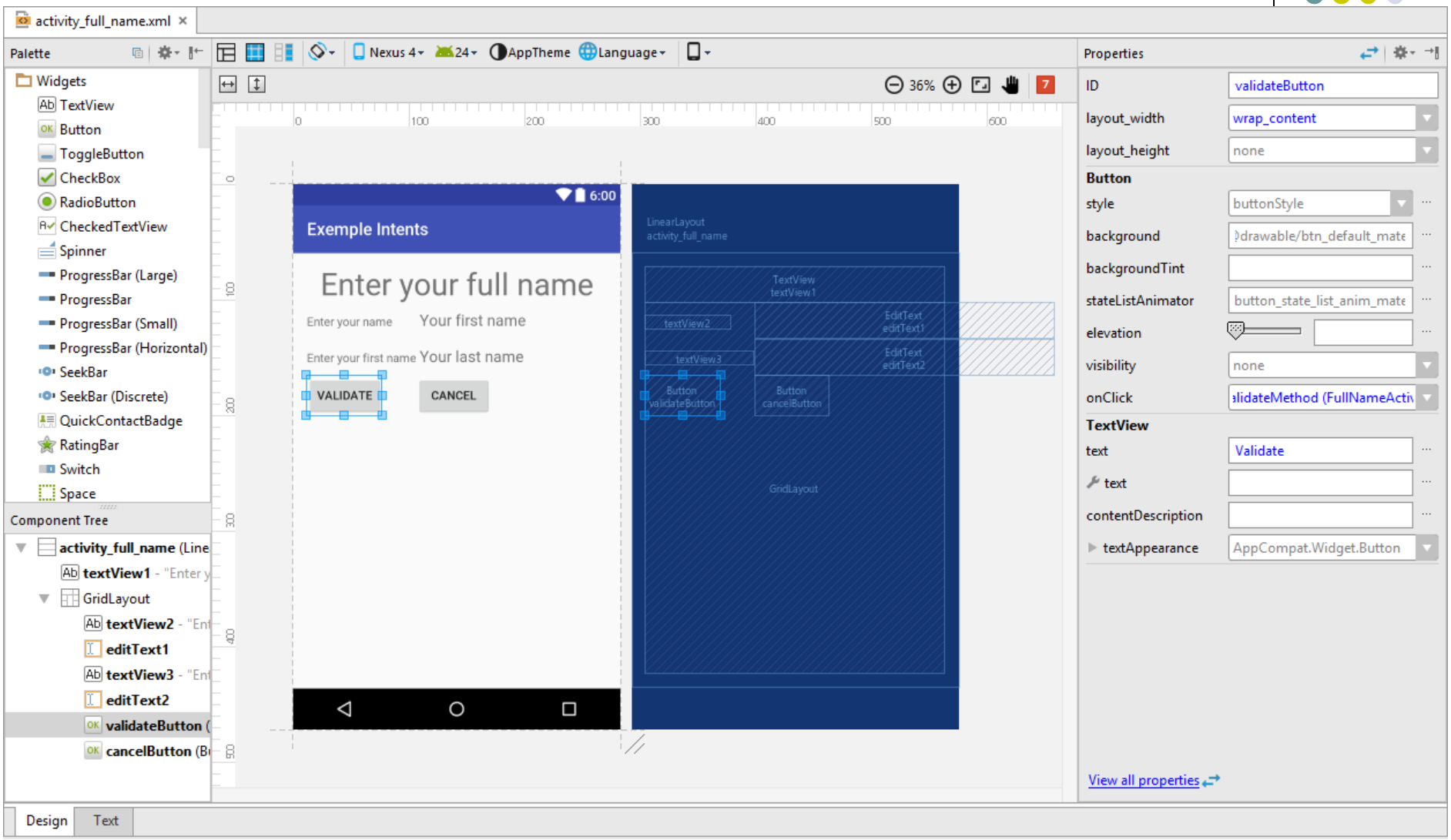
MainActivity.java

```
1  package com.example.toh.exempleintents;
2
3  import ...
8
9  public class MainActivity extends AppCompatActivity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_main);
15     }
16
17     public void loadFullNameActivity(View v) {
18         Intent myIntent = new Intent(MainActivity.this, FullNameActivity.class);
19         startActivityForResult(myIntent, 1);
20     }
21
22     @Override
23     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
24         if (requestCode == 1) {
25             if (resultCode == RESULT_CANCELED) {
26                 Toast.makeText(MainActivity.this, "You have cancelled !!!", Toast.LENGTH_LONG).show();
27             } else {
28                 String firstName = data.getStringExtra("fisrtName");
29                 String lastName = data.getStringExtra("lastName");
30                 Toast.makeText(MainActivity.this, "You have entrered as fisrt name :" +
31                         firstName + "  and as last name: " + lastName + "!!!", Toast.LENGTH_LONG).show();
32             }
33         }
34     }
35 }
```

FullName Activity



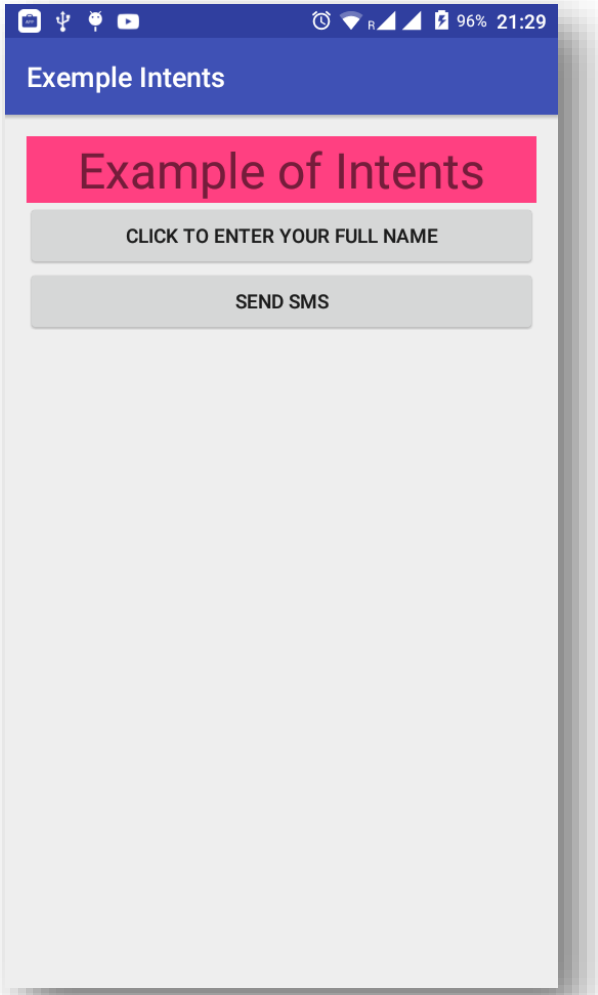
FullName Activity

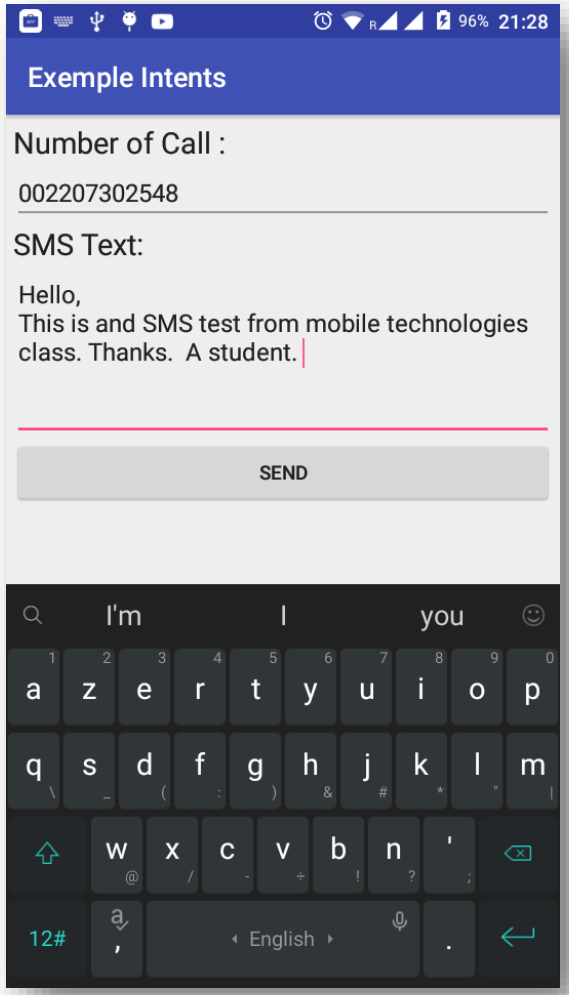


FullName Activity

FullNameActivity.java

```java
package com.example.toh.exempleintents;

import ...

public class FullNameActivity extends AppCompatActivity {
    EditText firstName, lastName;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_full_name);
        firstName = (EditText) findViewById(R.id.editText1);
        lastName = (EditText) findViewById(R.id.editText2);
    }

    public void validateMethod(View v) {
        if (firstName.getText().toString().trim().length() > 0 && lastName.getText().toString().trim().length() > 0) {
            getIntent().putExtra("fisrtName", firstName.getText().toString());
            getIntent().putExtra("lastName", lastName.getText().toString());
            setResult(RESULT_OK, getIntent());
            finish();
        }
    }

    public void cancelMethod(View v) {
        setResult(RESULT_CANCELED, getIntent());
        finish();
    }

}
```



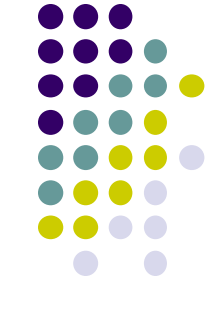


Example 2

This part aims to introduce of Android API to send and receive SMS. The Android SDK supports sending SMS / MMS in two ways:

• Summon via implicit Intents installed SMS client application. This method is to invoke SMS applications pre-installed on every Android device.

• Send SMS directly using the API SmsManager.



activity_sms_call.xml

LinearLayout | TextView

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/linearLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="5dp" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Number of Call : "
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/mobileNumber"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="Destination number"
        android:inputType="phone" >
    </EditText>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SMS Text: "
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/smsBody"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="top"
        android:hint="Write your SMS here"
        android:inputType="textMultiLine"
        android:lines="5" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:weightSum="10" >

        <Button
            android:id="@+id/send"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="5"
            android:text="Send"
            android:onClick="sendMethod" />
    </LinearLayout>

</LinearLayout>
```

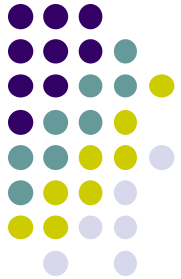
SmsCallActivity.java

```java
package com.example.toh.exempleintents;

import ...

public class SmsCallActivity extends AppCompatActivity {
    private EditText phoneNo;
    private EditText messageBody;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sms_call);
        phoneNo = (EditText) findViewById(R.id.mobileNumber);
        messageBody = (EditText) findViewById(R.id.smsBody);

    }

    public void sendMethod(View v) {
        String number = phoneNo.getText().toString();
        String sms = messageBody.getText().toString();
        try {
            SmsManager smsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(number, null, sms, null, null);
            Toast.makeText(getApplicationContext(), "SMS send successfully!", Toast.LENGTH_LONG).show();
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(),
                    "Failed to send the SMS or Call, Try Again!",
                    Toast.LENGTH_LONG).show();
            e.printStackTrace();
        }
    }
}
```

MainActivity.java

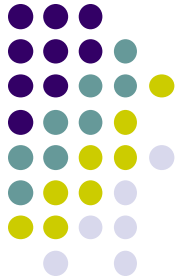
```
import ...

public class MainActivity extends AppCompatActivity {

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

    public void loadFullNameActivity(View v) {
        Intent myIntent = new Intent(MainActivity.this, FullNameActivity.class);
        startActivityForResult(myIntent, 1);
    }

    public void loadSmsCallActivity(View v){
        Intent myIntent = new Intent(MainActivity.this, SmsCallActivity.class);
        startActivityForResult(myIntent, 2);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == 1) {
            if (resultCode == RESULT_CANCELED) {
                Toast.makeText(MainActivity.this, "You have cancelled !!!", Toast.LENGTH_LONG).show();
            } else {
                String firstName = data.getStringExtra("fisrtName");
                String lastName = data.getStringExtra("lastName");
                Toast.makeText(MainActivity.this, "You have entrered as fisrt name :" +
                        firstName + "  and as last name: " + lastName + "!!!", Toast.LENGTH_LONG).show();
            }
        }
    }
}
```

Predefined Intentions



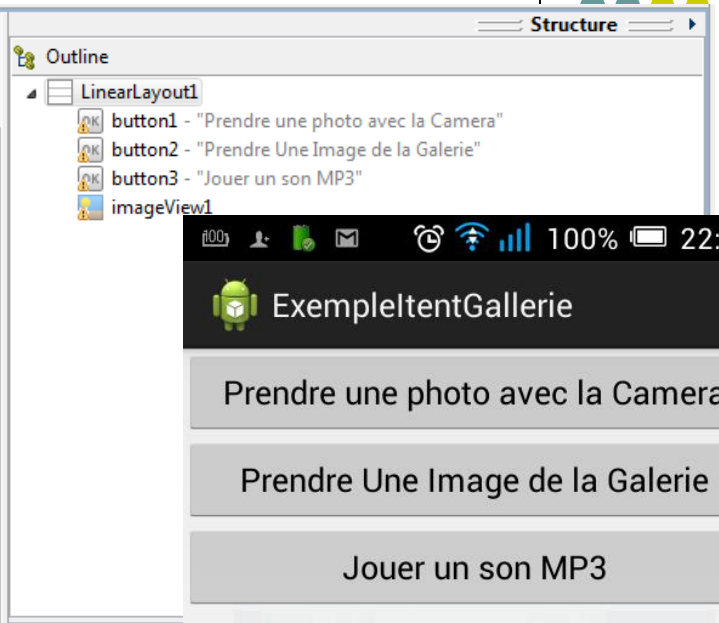
- **ACTION_MAIN**: main action
- **ACTION_VIEW**: visualize data
- **ACTION_ATTACH_DATAT**: data attachment
- **ACTION_EDIT**: Data Edition
- **ACTION_PICK**: Choose a data directory
- **ACTION_CHOOSER:** choice of menu for the user - EXTRA_INTENT contains the original Intent, EXTRA_TITLE the menu title
- **ACTION_GET_CONTENT**: get content following a MIME type
- **ACTION_SEND**: sent a message (EXTRA_TEXT | EXTRA_STREAM) to an unspecified recipient

Predefined Intents



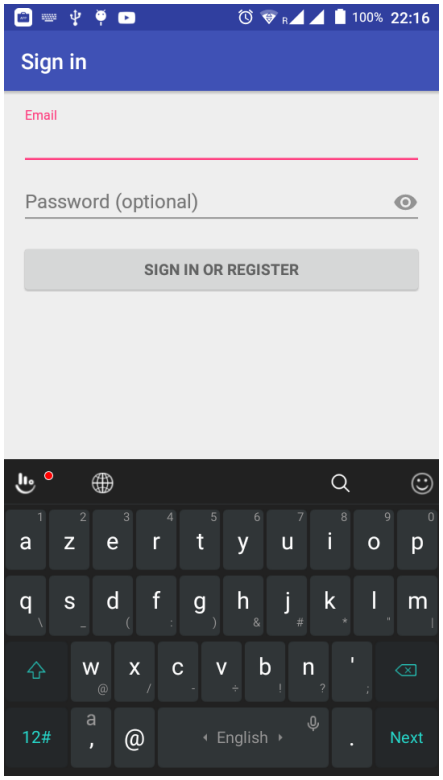
- **ACTION_SEND_TO**: it specifies the recipient in the URI
- **ACTION_INSERT**: adding a blank item in the directory specified by the URI
- **ACTION_DELETE**: it removes the element designated by the URI
- **ACTION_PICK_ACTIVITY**: selection menu according EXTRA_INTENT but do not start the activity
- **ACTION_SEARCH**: search etc ...

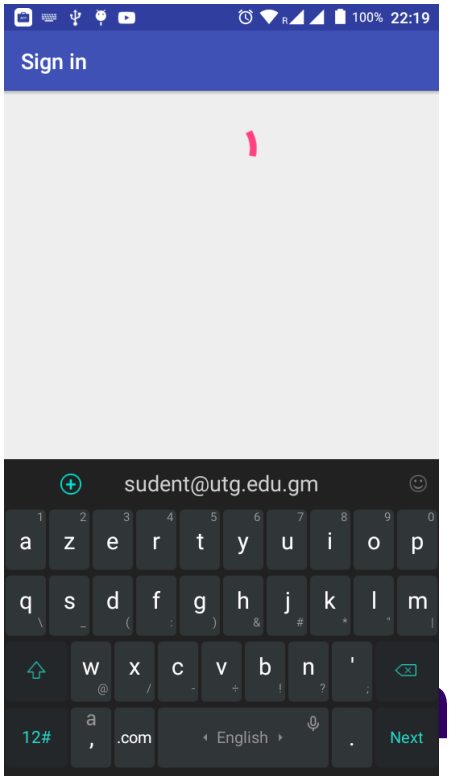
Predefined Intent

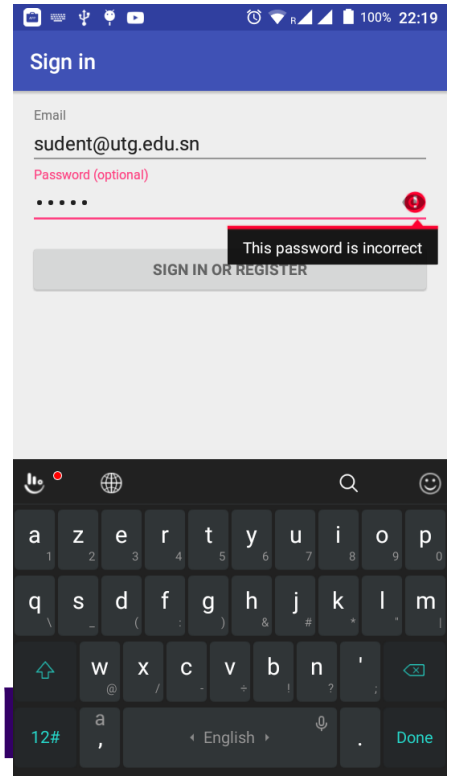


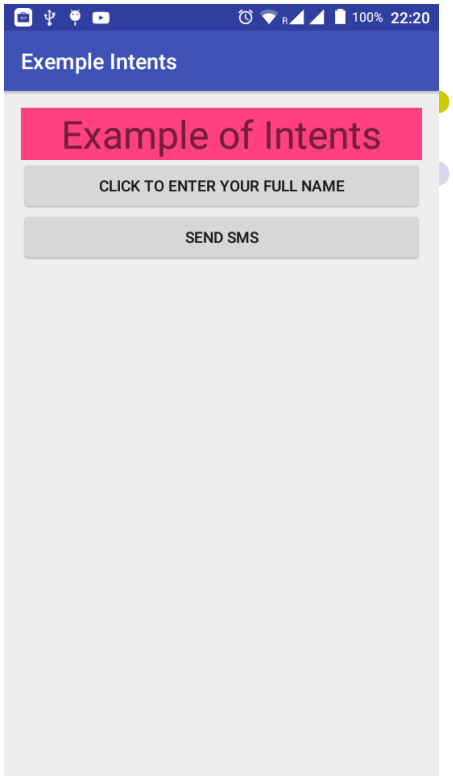
```
16 public class MainActivity extends Activity {
17     Button galerieButton, cameraButton, mp3Button;
18     ImageView myImageView;
19 
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22         super.onCreate(savedInstanceState);
23         setContentView(R.layout.activity_main);
24         myImageView = (ImageView) findViewById(R.id.imageView1);
25         cameraButton = (Button) findViewById(R.id.button1);
26         cameraButton.setOnClickListener(new OnClickListener() {
27             @Override
28             public void onClick(View arg0) {
29                 Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
30                 startActivityForResult(intent, 1);
31             }
32         });
33         galerieButton = (Button) findViewById(R.id.button2);
34         galerieButton.setOnClickListener(new OnClickListener() {
35             @Override
36             public void onClick(View arg0) {
37                 startActivityForResult(
38                         new Intent(Intent.ACTION_PICK,
39                                 android.provider.MediaStore.Images.Media.INTERNAL_CONTENT_URI),2);
40             }
41         });
42         mp3Button = (Button) findViewById(R.id.button3);
43         mp3Button.setOnClickListener(new OnClickListener() {
44             @Override
45             public void onClick(View arg0) {
46                 Intent intent = new Intent();
47                 intent.setType("audio/mp3");
48                 intent.setAction(Intent.ACTION_GET_CONTENT);
49                 startActivityForResult(Intent.createChooser(intent, "Open Audio (mp3) file"),3);
50             }
51         });
52     }
```

```
54    public void onActivityResult(int requestCode, int resultCode, Intent intent) {
55        super.onActivityResult(requestCode, resultCode, intent);
56        switch (requestCode) {
57        case 1:
58            if (resultCode == Activity.RESULT_OK) {
59                Bitmap bitmapImage = (Bitmap) intent.getExtras().get("data");
60                myImageView.setImageBitmap(bitmapImage);
61                myImageView.setVisibility(ImageView.VISIBLE);
62            }
63            break;
64
65        case 2:
66            if (resultCode == Activity.RESULT_OK) {
67                Uri mImageCaptureUri = intent.getData();
68                myImageView.setImageURI(mImageCaptureUri);
69                myImageView.setVisibility(ImageView.VISIBLE);
70            }
71            break;
72
73        case 3:
74            if (resultCode == Activity.RESULT_OK) {
75                Uri audioFileUri = intent.getData();
76                MediaPlayer myMediaPlayer = MediaPlayer.create(this, audioFileUri);
77                myMediaPlayer.start();
78
79                // info.setText(audioFileUri.getPath());
80            }
81            break;
82        }
83    }
```



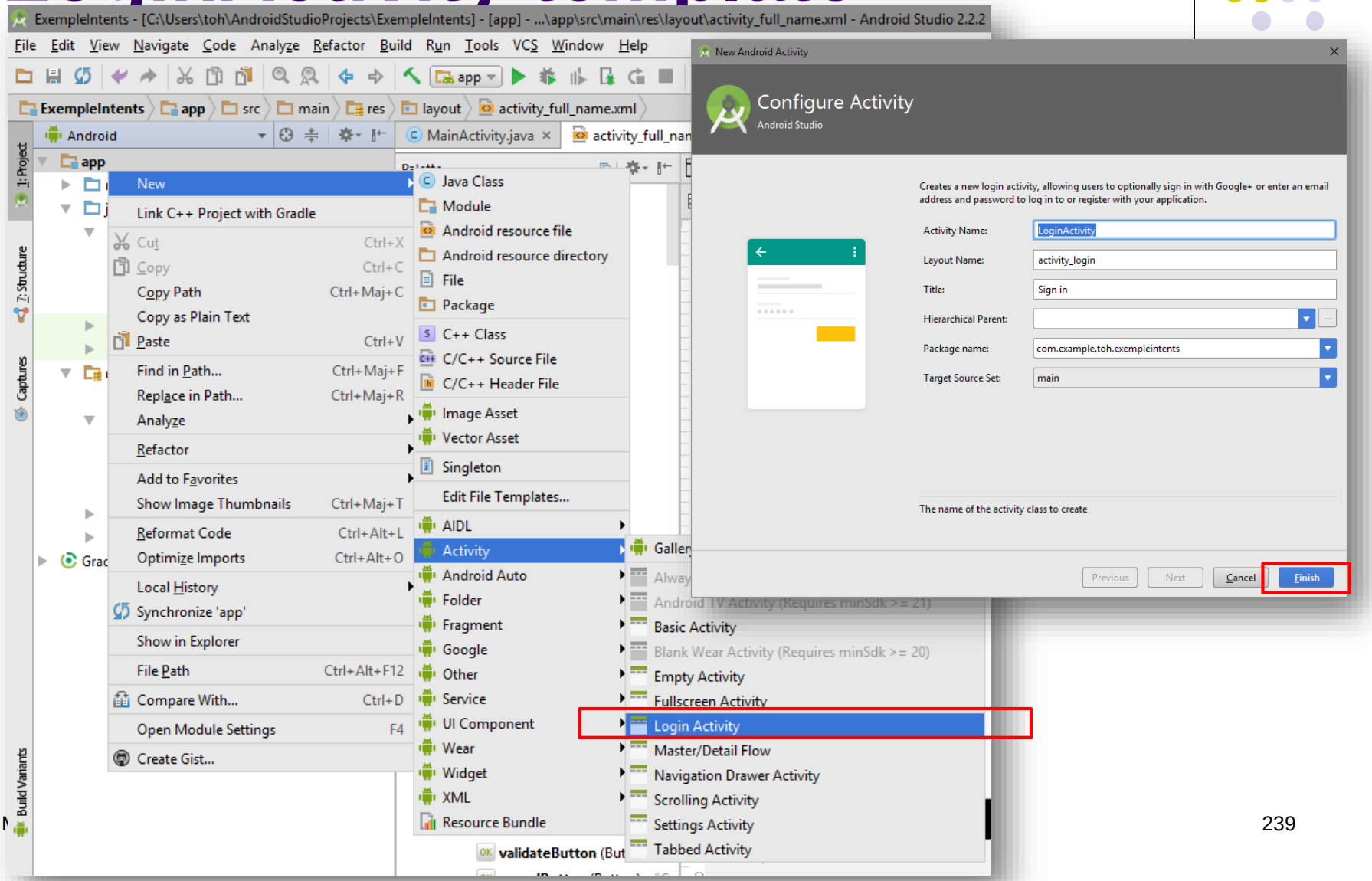




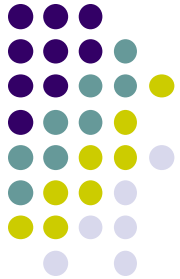


LoginActivity before accessing to main screen

1. Create new Activity using LoginActivity template



2. Modifying LoginActivity.java



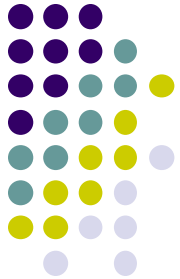
1

```
47      /**
48       * A dummy authentication store containing known user names and passwords.
49       * TODO: remove after connecting to a real authentication system.
50       */
51      private static final String[] DUMMY_CREDENTIALS = new String[]{
52              "sudent@utg.edu.gm:madsi", "login@utg.edu.gm:pass"
53      };
```

2

```
331     @Override
332     protected void onPostExecute(final Boolean success) {
333         mAuthTask = null;
334         showProgress(false);
335
336         if (success) {
337             finish();
338             Intent myIntent=new Intent(LoginActivity.this, MainActivity.class);
339             startActivity(myIntent);
340
341         } else {
342             mPasswordView.setError(getString(R.string.error_incorrect_password));
343             mPasswordView.requestFocus();
344         }
345     }
```

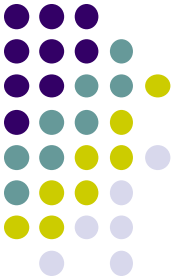
2. Modifying LoginActivity.java



```
308         @Override
309         protected Boolean doInBackground(Void... params) {
310             // TODO: attempt authentication against a network service.
311
312             try {
313                 // Simulate network access.
314                 Thread.sleep(2000);
315             } catch (InterruptedException e) {
316                 return false;
317             }
318
319             for (String credential : DUMMY_CREDENTIALS) {
320                 String[] pieces = credential.split(":");
321                 if (pieces[0].equals(mEmail)) {
322                     // Account exists, return true if the password matches.
323                     return pieces[1].equals(mPassword);
324                 }
325             }
326
327             // TODO: register the new account here.
328             return false;
```

3

3. Set LoginActivity as laucher activity by changin Manisfest



AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.toh.exempleintents">
4
5      <uses-permission android:name="android.permission.SEND_SMS" />
6
7      <!-- To auto-complete the email text field in the login form with the user's emails -->
8      <uses-permission android:name="android.permission.GET_ACCOUNTS" />
9      <uses-permission android:name="android.permission.READ_PROFILE" />
10     <uses-permission android:name="android.permission.READ_CONTACTS" />
11
12     <application
13         android:allowBackup="true"
14         android:icon="@mipmap/ic_launcher"
15         android:label="Exemple Intents"
16         android:supportsRtl="true"
17         android:theme="@style/AppTheme">
18         <activity android:name=".MainActivity">
19             <intent-filter>
20                 <action android:name="android.intent.action.MAIN" />
21
22                 <category android:name="android.intent.category.LAUNCHER" />
23             </intent-filter>
24         </activity>
25         <activity android:name=".FullNameActivity" />
26         <activity android:name=".SmsCallActivity" />
27         <activity
28             android:name=".LoginActivity"
29             android:label="@string/title_activity_login">
30
31         </activity>
32     </application>
33
34 </manifest>
```

Move this block to line number 30

Développement d'applications mobiles

ANDROID- FILES

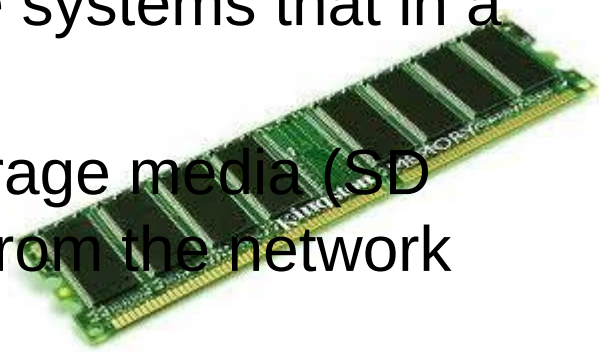
M2 RT/GLAR

PARTIE 8

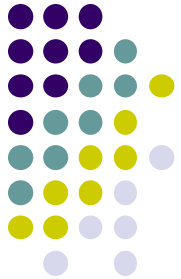
Pr. El hadji M. NGUER

Files on Android

- Android uses the same constructs file systems that in a typical Java application.
- Files can be stored in any type of storage media (SD Card, ..). They can also be obtained from the network (as discussed later).
- Files stored in the device memory, are together with other application resources (such as icons, images, music, ...).
- We will call this type: resource files.



Files on Android



- Data storage options are as follows :

  1. Shared Preferences to save the data in the form of key-value pairs.

  2. Internal Storage Media

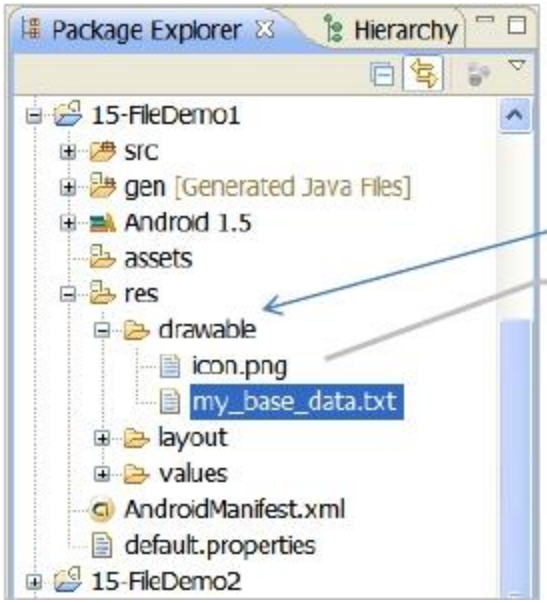
  3. External storage support

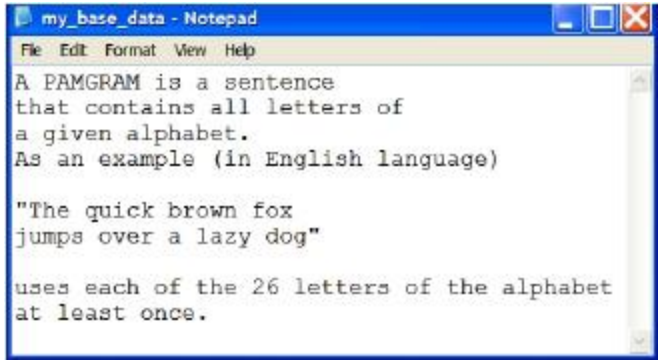
  4. SQLiteDatabases

  5. Using the network for a backup on the web for example

Files on Android

```
InputStream is = this.getResources()
                    .openRawResource(R.drawable.my_base_data);
```





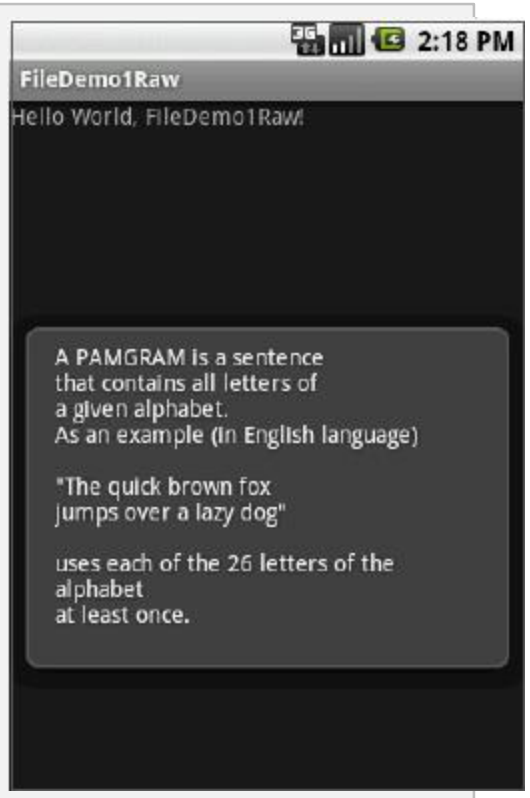
La cigüeña tocaba cada vez mejor el saxofón y el búho pedía kiwi y queso.

5

Les Fichiers sous Android

Exemple 0: Lire un Fichier Resource

```
//reading an embedded RAW data file
package sn.ugb.MaRT.fichiers;
import android.app.Activity;
import android.os.Bundle;
import android.widget.Toast;
import java.io.*;

public class FileDemo1Raw extends Activity {

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    try {
        PlayWithRawFiles();
    } catch (IOException e) {
        Toast.makeText(getApplicationContext(),
                       "Problems: " + e.getMessage(), 1).show();
    }
}// onCreate
```



Android Files

Example 1: Reading a Resource File (see previous figure)

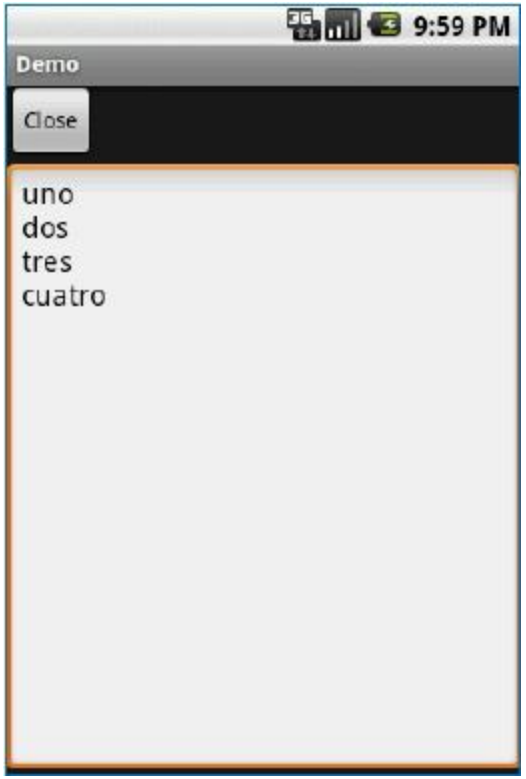
```
public void PlayWithRawFiles() throws IOException {
        String str="";
        StringBuffer buf = new StringBuffer();

        InputStream is = this.getResources()
                        .openRawResource(R.drawable.my_base_data);

        BufferedReader reader = new BufferedReader(
                                new InputStreamReader(is));
        if (is!=null) {
            while ((str = reader.readLine()) != null) {
                buf.append(str + "\n" );
            }
        }
        is.close();
        Toast.makeText(getBaseContext(),
                        buf.toString(), Toast.LENGTH_LONG).show();
}// PlayWithRawFiles

} // FilesDemo1
```

Android Files

Example2: (Internal Storage) Read/Write an Internal File.

In this example an application collects data from the UI and saves it to a persistent data file into the (limited) internal Android System Space area.

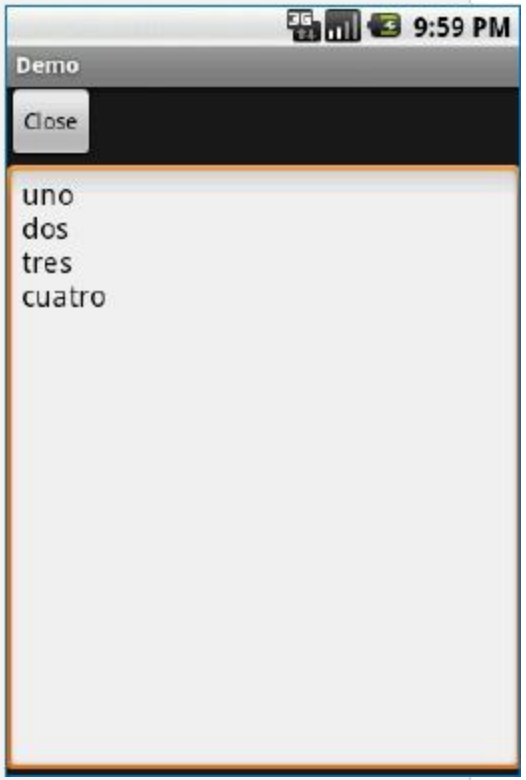
Next time the application is executed the Resource file is read and its data is shown in the UI



Android Files

Example2: Grab from screen, save to file, retrieve from file.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <Button android:id="@+id/close"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Close" />

    <EditText
        android:id="@+id/editor"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:singleLine="false"
        android:gravity="top"
        />

</LinearLayout>
```



Android Files

Example2: Grab from screen, save to file, retrieve from file.

```
package cis493.demo;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;


public class Demo extends Activity {

    private final static String NOTES="notes.txt";
    private EditText editor;
```

Android Files

Example2: Grab from screen, save to file, retrieve from file.

```
@Override
public void onCreate(Bundle icicle) {
     super.onCreate(icicle);

     setContentView(R.layout.main);
     editor=(EditText)findViewById(R.id.editor);

     Button btn=(Button)findViewById(R.id.close);

     btn.setOnClickListener(new Button.OnClickListener() {
          public void onClick(View v) {
            finish();
          }
     });

}//onCreate
```

Android Files

Example2: Grab from screen, save to file, retrieve from file.

```
public void onResume() {
    super.onResume();
    try {
        InputStream in=openFileInput(NOTES);
        if (in!=null) {
            InputStreamReader tmp=new InputStreamReader(in);
            BufferedReader reader=new BufferedReader(tmp);
            String str;
            StringBuffer buf=new StringBuffer();

            while ((str = reader.readLine()) != null) {
                buf.append(str+"\n");
            }
            in.close();
            editor.setText(buf.toString());
        }//if
    }
    catch (java.io.FileNotFoundException e) {
        // that's OK, we probably haven't created it yet
    }
    catch (Throwable t) {
        Toast.makeText(this, "Exception: "+ t.toString(), 2000).show();
    }

}
```

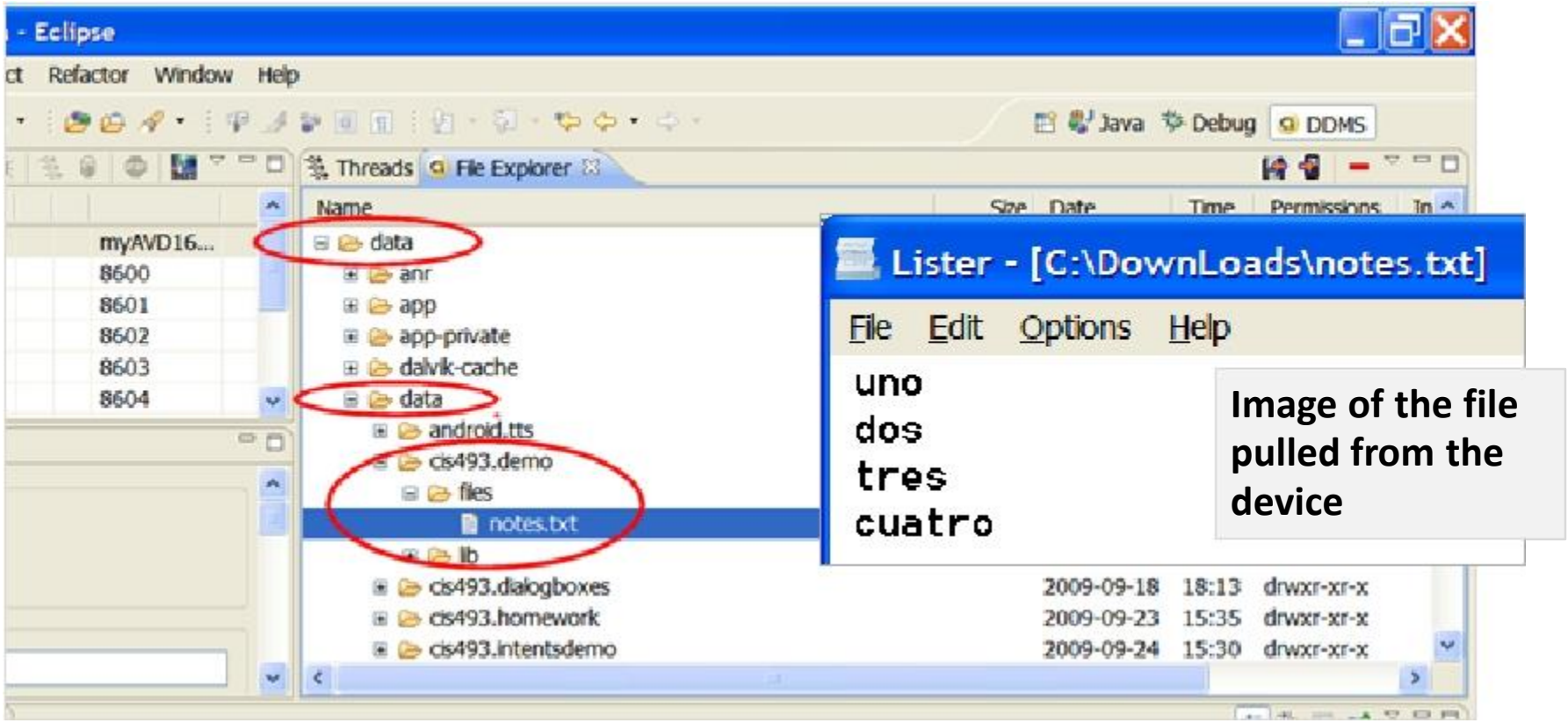
Android Files

Example2: Grab from screen, save to file, retrieve from file.

```
public void onPause()  {
    super.onPause();

    try {
        OutputStreamWriter out=
        new OutputStreamWriter(openFileOutput(NOTES,  0));

        out.write(editor.getText().toString());
        out.close();
    }
    catch (Throwable t) {
        Toast.makeText(this, "Exception: "+ t.toString(), 2000).show();
    }
}


}//class
```

Android Files

File is stored in the phone's memory under: /data/data/app/files

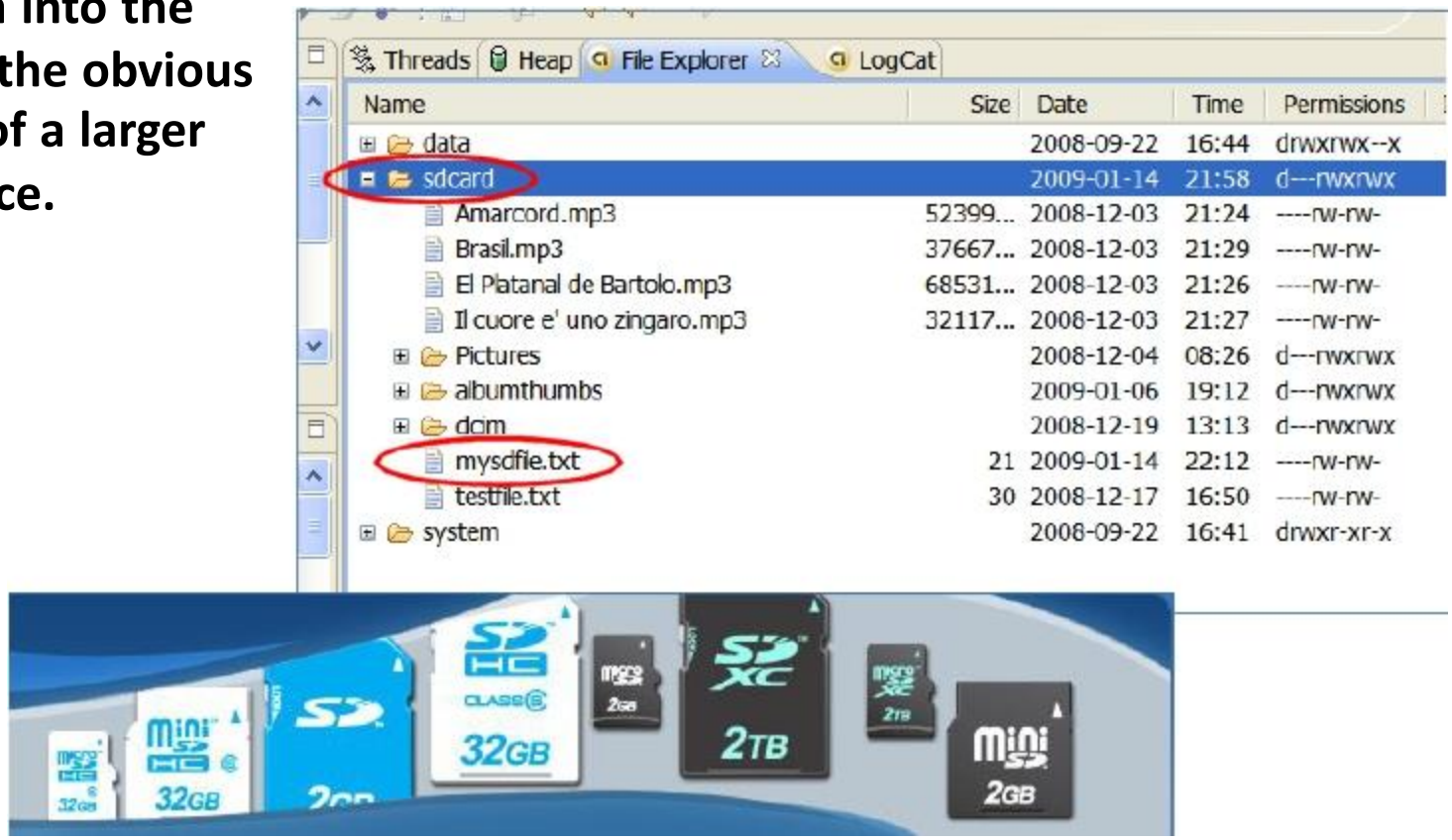


Android Files

Example 3: (External Storage)

Reading/Writing to the External Device's SD card.

Storing data into the SD card has the obvious advantage of a larger working space.



Android Files

WARNING: Writing to the Device's SD card.

Since SDK1.6 it is necessary to request permission to write to the SD card. Add the following clause to your AndroidManifest.xml

```
<uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE">
</uses-permission>
```

Android Files

Example 3: Reading/Writing to the Device's SD card.

Assume the SD card in this example has been named *sdcard*. We will use the Java.io.File class to designate the file's path. The following fragment illustrates the code strategy for output files.

```
File myFile = new File("/sdcard/mysdfile.txt");
myFile.createNewFile();

FileOutputStream fOut = new FileOutputStream(myFile);
OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
myOutWriter.append(txtData.getText());
myOutWriter.close();
fOut.close();
```

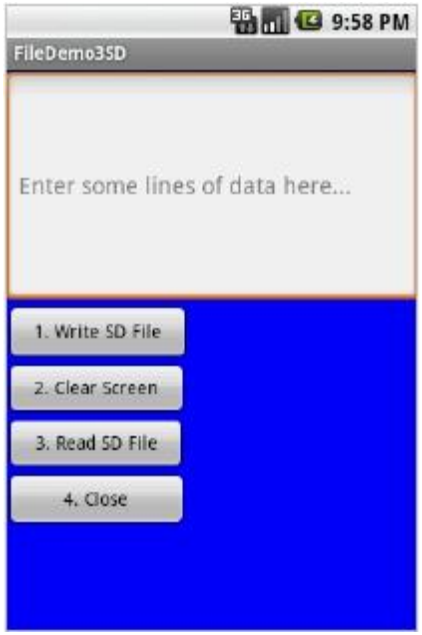
Android Files

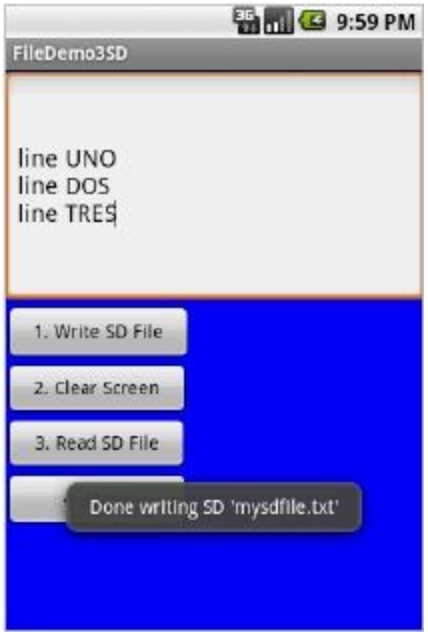
Example 3: Reading/Writing to the Device's SD card.

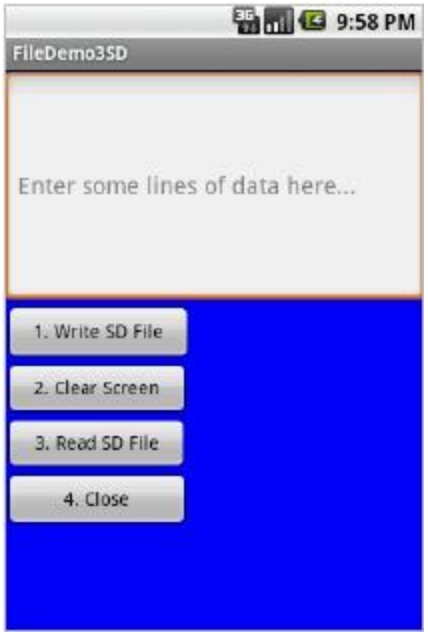
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android=http://schemas.android.com/apk
/res/android
android:id="@+id/widget28"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ff0000ff"
android:orientation="vertical"
>
<EditText
android:id="@+id/txtData"
android:layout_width="fill_parent"
android:layout_height="180px"
android:text="Enter some data here ..."
android:textSize="18sp" />

<Button
android:id="@+id/btnWriteSDFile"
android:layout_width="143px"
android:layout_height="44px"
android:text="1. Write SD File" />

<Button
android:id="@+id/btnClearScreen"
android:layout_width="141px"
android:layout_height="42px"
android:text="2. Clear Screen" />

<Button
android:id="@+id/btnReadSDFile"
android:layout_width="140px"
android:layout_height="42px"
android:text="3. Read SD File" />

<Button
android:id="@+id/btnClose"
android:layout_width="141px"
android:layout_height="43px"
android:text="4. Close" />

</LinearLayout>
```

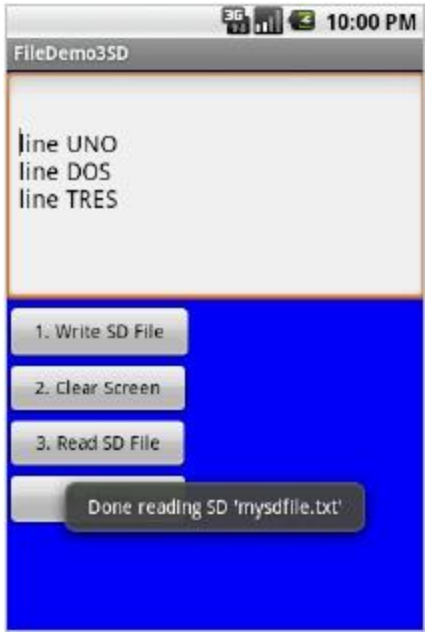
Android Files

Example 3: Reading/Writing to the Device's SD card.



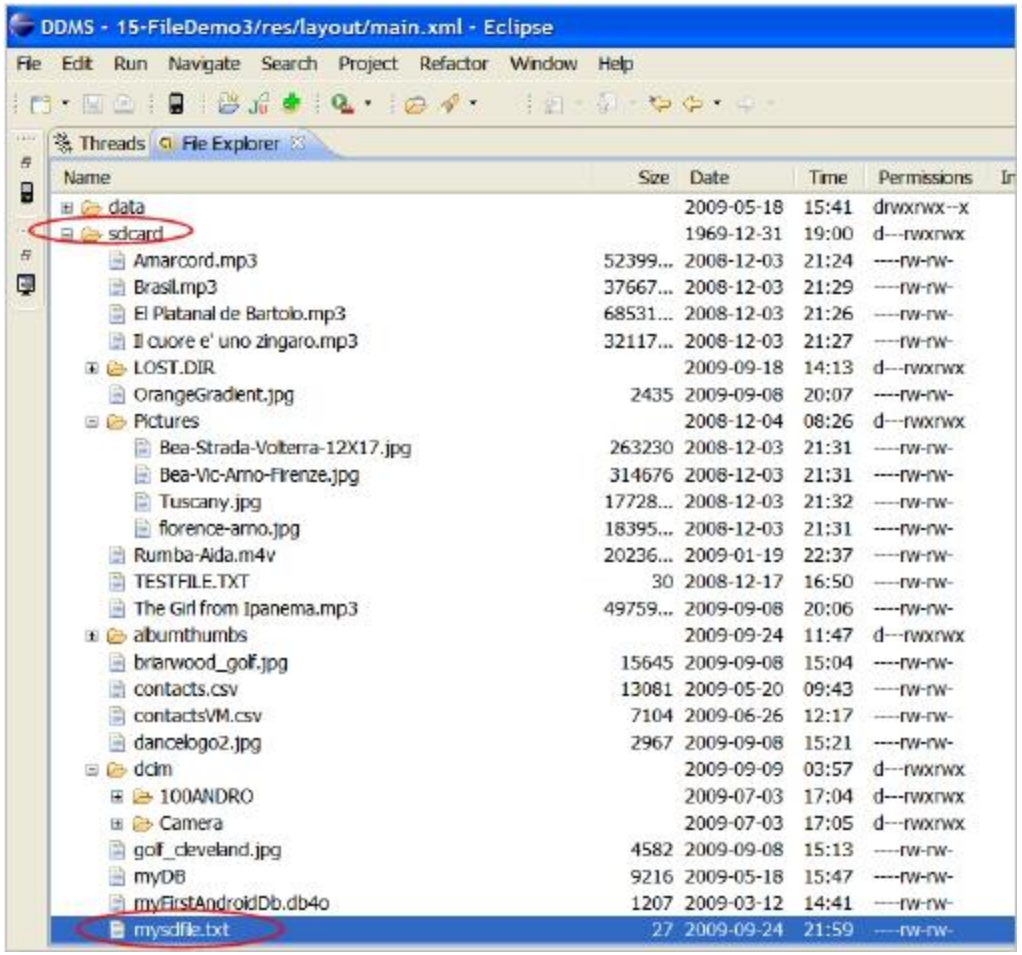






Android Files

Example 3: Reading/Writing to the Device's SD card.



Using DDMS *File Explorer* panel to inspect the SD card.

Android Files

Example 3: Reading/Writing to the Device's SD card.

```
package cis493.filedemo;
import java.io.*;
import android.app.Activity;
import android.os.Bundle;
import android.view.*;
import android.view.View.OnClickListener;
import android.widget.*;

public class FileDemo3SD extends Activity {
// GUI controls
EditText txtData;
Button btnWriteSDFile;
Button btnReadSDFile;
Button btnClearScreen;
Button btnClose;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    // bind GUI elements with local controls
    txtData = (EditText) findViewById(R.id.txtData);

    txtData.setHint("Enter some lines of data here...");
```

Android Files

Example 3: Reading/Writing to the Device's SD card.

```
btnWriteSDFile = (Button) findViewById(R.id.btnWriteSDFile);
btnWriteSDFile.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
    // write on SD card file data from the text box
    try {
        File myFile = new File("/sdcard/mysdfile.txt");
        myFile.createNewFile();
        FileOutputStream fOut = new FileOutputStream(myFile);
        OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
        myOutWriter.append(txtData.getText());
        myOutWriter.close();
        fOut.close();
        Toast.makeText(getBaseContext(),
                "Done writing SD 'mysdfile.txt'",
        Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        } Toast.makeText(getBaseContext(),
                e.getMessage(), Toast.LENGTH_SHORT).show();


    }// onClick
}); // btnWriteSDFile
```

Android Files

Example 3: Reading/Writing to the Device's SD card.

```
btnReadSDFile = (Button) findViewById(R.id.btnReadSDFile);
btnReadSDFile.setOnClickListener(new OnClickListener() {
 @Override
 public void onClick(View v) {
 // write on SD card file data from the text box
 try {
  File myFile = new File("/sdcard/mysdfile.txt");
  FileInputStream fIn = new FileInputStream(myFile);
  BufferedReader myReader = new BufferedReader(new InputStreamReader(fIn));
  String aDataRow = "";
  String aBuffer = "";
  while ((aDataRow = myReader.readLine()) != null) {
     aBuffer += aDataRow + "\n";
  }
  txtData.setText(aBuffer);
  myReader.close();
  Toast.makeText(getBaseContext(),
       "Done reading SD 'mysdfile.txt'", 1).show();
  } catch (Exception e) {
      Toast.makeText(getBaseContext(), e.getMessage(), 1).show();
  }
 }// onClick
}); // btnReadSDFile
```

Android Files

Example 3: Reading/Writing to the Device's SD card.

```
btnClearScreen = (Button) findViewById(R.id.btnClearScreen);
btnClearScreen.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        // clear text box
        txtData.setText("");
    }
}); // btnClearScreen

btnClose = (Button) findViewById(R.id.btnClose);
btnClose.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        // clear text box
        finish();
    }
}); // btnClose

}// onCreate


}// class
```

Android Files

Example 3: Reading/Writing to the Device's SD card. You may also use the Scanner/PrintWriter classes, as suggested below:

```
private void testScannerFiles(){
// Add to manifest the following permission request
// <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
     try {
          String SDcardPath = Environment.getExternalStorageDirectory().getPath();
          String mySDFileName = SDcardPath + "/" + "mysdfiletest.txt";
          tvMessage.setText("Writing to: " + mySDFileName);
          PrintWriter outfile= new PrintWriter( new FileWriter(mySDFileName) );
               outfile.println("Hola Android");
               outfile.println("Adios Android");
               outfile.println(new Date().toString());
          outfile.close();
          // read SD-file,show records.
          Scanner infile= new Scanner(new FileReader(mySDFileName));
          String inString= "\n\nReading from: " + mySDFileName + "\n";
          while(infile.hasNextLine()) {
               inString += infile.nextLine() + "\n";
          }
          tvMessage.append(inString);
          infile.close();
     } catch (FileNotFoundException e) {
          tvMessage.setText( "Error: " + e.getMessage());
     } catch (IOException e) {
          tvMessage.setText( "Error: " + e.getMessage());

     }
}
```

Android Files

Example 4: Some more ideas on using the Scanner/PrintWriter classes.

```
// writing
FileOutputStream fos = openFileOutput("XYZ",
Context.MODE_PRIVATE);

PrintWriter outfile= new PrintWriter( fos );
outfile.println("Hola Android");
outfile.close();


// reading
 InputStream is = openFileInput("XYZ");
 Scanner infile= new Scanner(is);
 String inString= "";
 while(infile.hasNextLine()) {
    inString = infile.nextLine();
 }
```

Files

Accessing the SD card

```
String spPath = “”;

sdPath = Environment.getExternalStorageDirectory()
            .getAbsolutePath()
            + “/myFileName.txt” ;
```

Développement d'applications mobiles

ANDROID- NOTIFICATIONS

M2 RT/GLAR

PARTIE 9

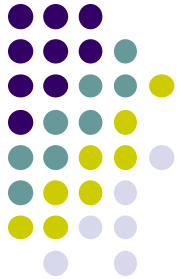
Pr. El hadji M. NGUER

Notifications: La classe Toast

- Texte qui apparaît en premier plan puis disparaît au bout d'un temps donné

- ***Création d'un Toast***
  - Toast.makeText(Context, String, int) renvoie l'objet de classe Toast créé.
  - Le premier paramètre est l'activité
  - Le deuxième paramètre est le message à afficher
  - Le dernier paramètre indique la durée d'affichage les seules valeurs possibles sont : Toast.LENGTH_SHORT (2 secondes) ou Toast.LENGTH_LONG (5 secondes).
- ***Positionnement d'un Toast***
  - setGravity(int, int, int) appelée avant l'affichage par show pour indiquer où s'affichera le message.
  - Le premier paramètre sert à placer le message par rapport à l'écran. Il peut prendre l'une des valeurs définies dans la classe Gravity soit : Gravity. (TOP, BOTTOM, LEFT, RIGHT, CENTER_VERTICAL, FILL_VERTICAL, CENTER_HORIZONTAL, FILL_ HORIZONTAL, CENTER, FILL).
  - Les deux paramètres suivants indiquent le décalage (en pixels).
- ***Affichage d'un Toast***
  - show() affiche le message pour la durée définie lors de sa création.

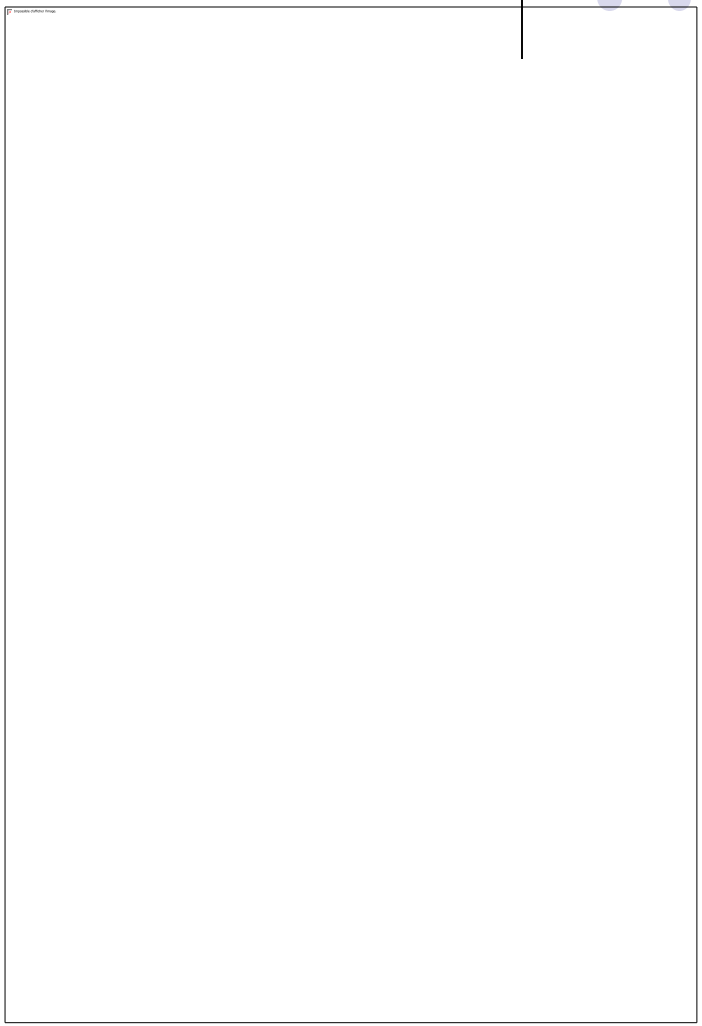
Notifications: La classe AlertDialog



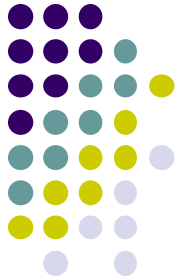
- Style plus classique des boîtes de dialogue. Un AlertDialog s'ouvre, prend le focus et reste affiché tant que l'utilisateur ne le ferme pas
- **Pour créer un AlertDialog:**
  - Utiliser la classe Builder offrant un ensemble de méthodes permettant de configurer un AlertDialog. Méthodes renvoie le Builder afin de faciliter le chaînage des appels.
  - À la fin, il suffit d'appeler la méthode show() de l'objet Builder pour afficher la boîte de dialogue.
- **Méthodes de configuration de Builder :**
  - setMessage() permet de définir le "corps" de la boîte de dialogue
  - setTitle() et setIcon() permettent de configurer le texte et/ou l'icône
  - setPositiveButton(), setNeutralButton() et setNegativeButton() permettent d'indiquer les boutons qui apparaîtront en bas de la boîte de dialogue, leur emplacement latéral (respectivement, à gauche, au centre ou à droite), leur texte et le code qui sera appelé lorsqu'on clique sur un bouton (en plus de refermer la boîte de dialogue).

Notifications: : Exemple de Toast

- Un **Toast est un message qui apparait et disparait**
- Il est impossible de savoir si l'utilisateur l'a vu ou pas
- Configuration:
  - Un **String contenant le message**
  - Une durée
    - **Toast.LENGTH_LONG ou Toast.LENGTH_SHORT**
- Il est aussi possible de donner une **View de votre choix en paramètre**

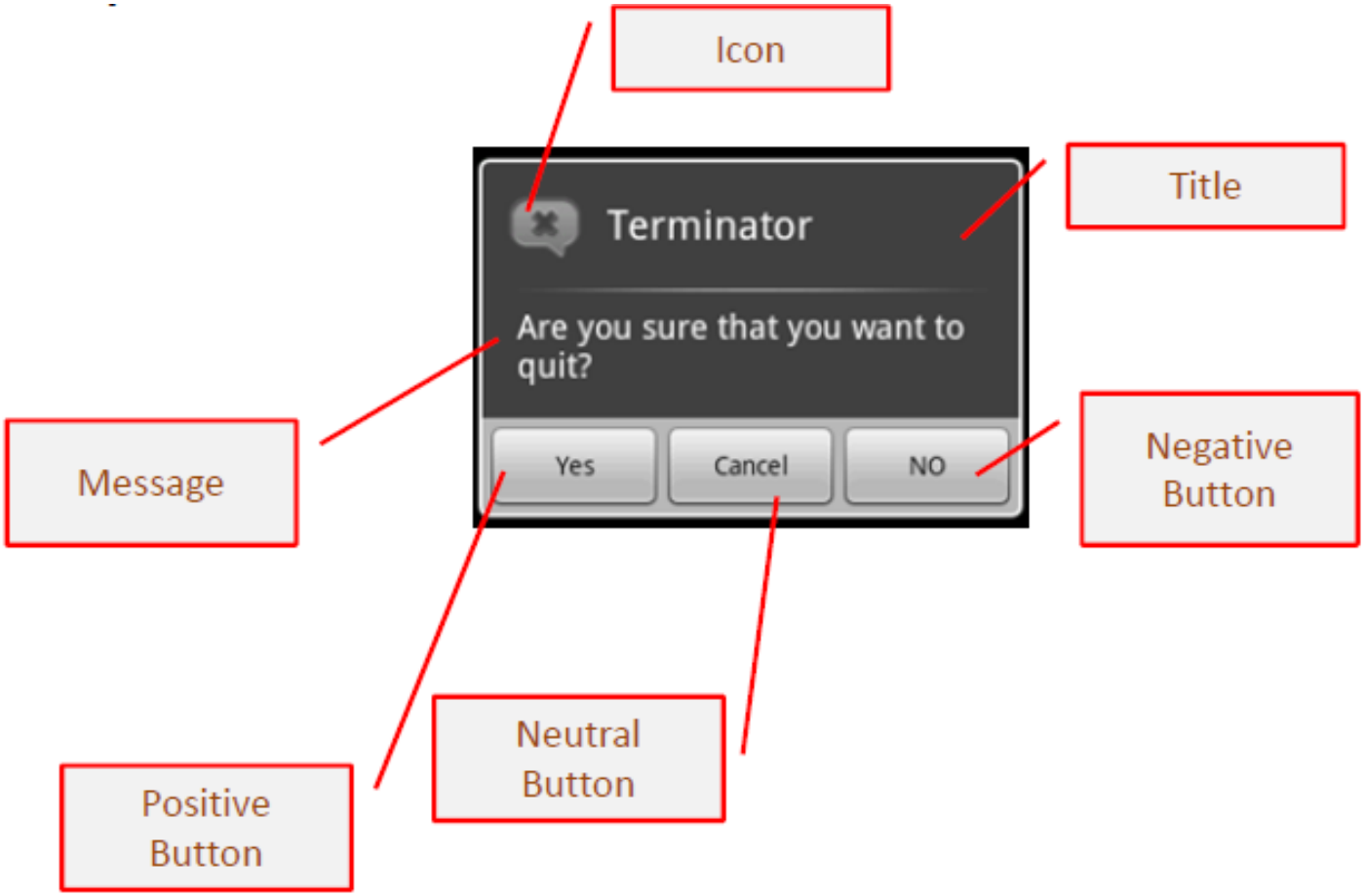


Toast : exemple



```
Toast.makeText(MainActivity.this,
    "C'est vous qui voyez!!!",
    Toast.LENGTH_SHORT).show();
```

Boites de dialogue



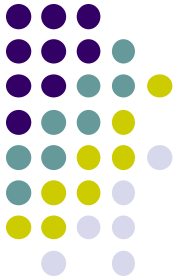
Exemple: AlertDialog

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <Button
        android:id="@+id/btnCliquer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="16dp"
        android:text="@string/cliquez_" />
    <EditText
        android:id="@+id/txtMsg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/btnCliquer"
        android:layout_alignBottom="@+id/btnCliquer"
        android:layout_alignParentRight="true"
        android:ems="10"
        android:hint="@string/cliquez_sur_le_bouton" />
</RelativeLayout>
```

Graphical Layout | activity_main.xml

Exemple: AlertDialog



activity_main.xml | MainActivity.java

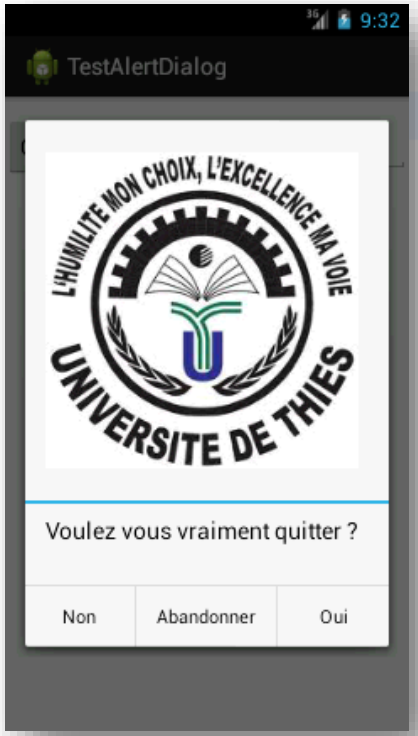
```
package com.example.testalertdialog;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity {
    Button cliquerButton;
    EditText txtMsg;
    String msg;
```

MainActivity.java

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtMsg = (EditText) findViewById(R.id.txtMsg);
        cliquerButton = (Button) findViewById(R.id.btnCliquer);
        cliquerButton.setOnClickListener(new OnClickListener() {
            public void onClick(View arg0) {
                AlertDialog diaBox = createDialogBox();
                diaBox.show();
                txtMsg.setText("Je suis là !!!");
            }
        });
    }
```

Exemple: AlertDialog

```
protected AlertDialog createDialogBox() {
    AlertDialog myQuittingDialogBox = new AlertDialog.Builder(this)
            // set message, title, and icon
            .setTitle("Terminator")
            .setMessage("Voulez vous vraiment quitter ?")
            .setIcon(R.drawable.logouthies)
            // set three option buttons
            .setPositiveButton("Oui",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                                int whichButton) {
                            msg = "Oui " + Integer.toString(whichButton);
                            txtMsg.setText(msg);
                        }
                    })// setPositiveButton
```



Exemple: AlertDialog

```
            .setNeutralButton("Abandonner",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                                int whichButton) {
                            // whatever should be done when answering
                            // "CANCEL" goes here
                            msg = "Abandonner "
                                    + Integer.toString(whichButton);
                            txtMsg.setText(msg);
                        }// OnClick
                    })// setNeutralButton
            .setNegativeButton("Non",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                                int whichButton) {
                            // whatever should be done when answering "NO"
                            // goes here
                            msg = "Non " + Integer.toString(whichButton);
                            txtMsg.setText(msg);
                        }
                    })// setNegativeButton
            .create();
    return myQuittingDialogBox;
}// createDialogBox
```

Développement d'applications mobiles

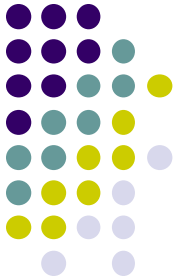
ANDROID- PERSISTENCE: SQLITE DB

M2 RT/GLAR

PARTIE 10

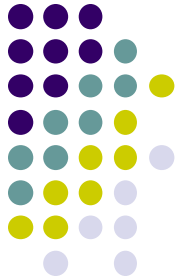
Pr. El hadji M. NGUER

SQLite



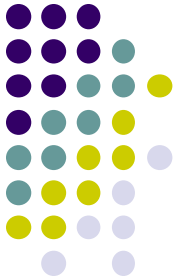
- Relational database engine
- Created by D.Richard Hipp
- Written in C (ANSI-C)
- Open source without restriction
- Multiplatform
- Most distributed in the world (Firefox, Skype, Google Gears)
- Embedded systems (iPhone, Symbian, Android, but not on Windows Phone)

SQLite



- Implements the SQL-92 standard (aka SQL2)
- And ACID properties
- embedded DBMS
- No rights management (single user)
- No configuration

But…



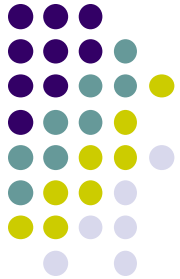
- Depending on the file system
- A database = a file
- Beware the limit on SD formatted in FAT 32
- Not own extension (".sqlite", ".db" is good practice)
- Backup option RAM (extension ": memory:")
- Objective: To replace the file system, but not RDBMS

Your library with SQLite

- The POJO Book
- An object that represents a record "book"
- Attributes, accessors, a stateless and stateful manufacturer

```
public class Book {

  // attributes
  private int id;
  private String title;
  private String isbn;
  private int nbPage;

  // accessors
  public String getTitle() {
    return title;
  }

  public void setTitle(String title) {
     this.title = title;
  }

  // etc.

  // constructor
  public Book(int id) {
    super(id);
  }

  public Book(int id, String title, String isbn, int nbPage) {
    this(id);
    this.title = title;
    this.isbn = isbn;
    this.nbPage = nbPage;
  }

}
```

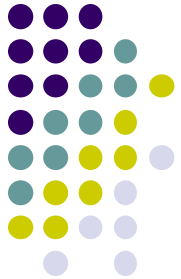
Creating the database



- It must provide a class that has responsibility for creating the database and update the schema

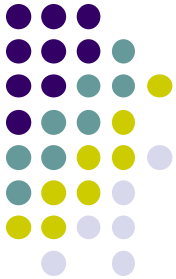
```
public class LibraryDBHelper extends SQLiteOpenHelper {

  @Override
  public void onCreate(SQLiteDatabase db) {
    // TODO Auto-generated method stub
  }

  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // TODO Auto-generated method stub
  }
}
```

Creating the database



- Abstract class : SQLiteOpenHelper
- A constructor: useful data
- Two abstract methods :
  - onCreate : responsible for creating the BD
  - onUpgrade : responsible for the schema update (in general, it erases everything and start again)

Constructor



- The Android context
- The filename
- The cursor creator (null by default)
- The version
- A handler in case of database corruption (default)

```
public static final String DB_NAME = "Library.db";
public static final int DB_VERSION = 5;

// constructor
public LibraryDBHelper(Context context) {
  super(context, DB_NAME, null, DB_VERSION);
}
```

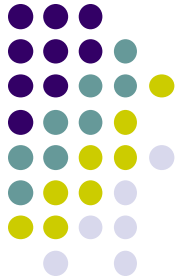
Creation and Upgrade

```
public class LibraryDBHelper extends SQLiteOpenHelper {

  public static final String DB_NAME = "Library.db";
  public static final int DB_VERSION = 5;

  public static String getQueryCreate() {
    return "CREATE TABLE Book ("
        + "id Integer PRIMARY KEY AUTOINCREMENT, "
        + "title Text NOT NULL, "
        + "isbn Text NOT NULL, "
        + "nbPage Integer NOT NULL"
        + ");";
  }

  public static String getQueryDrop() {
    return "DROP TABLE IF EXISTS Book;";
  }

  @Override
  public void onCreate(SQLiteDatabase db) {
    db.execSQL(getQueryCreate());
  }

  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL(getQueryDrop());
    db.execSQL(getQueryCreate());
  }
}
```

Creating tables

Recreate the tables

Finer management upgrade

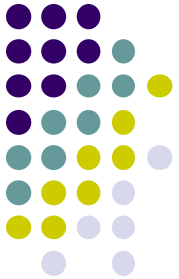


- Small steps up to the current version

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
  // Management by successive upgrades
  // newVersion is 4 !
  int delta = newVersion - oldVersion;
  if (delta >= 3) {
    // upgrate to version 2
  }

  if (delta >= 2) {
    // upgrate to version 3
  }

  if (delta >= 1) {
    // upgrate to version 4
  }
}
```

Manages changes step by step

Datasource



- Responsible class access to the database
- Responsible for creating the helper
- Open (open())/Close (close())/Give access to DB (getDB())
- Factory of different DAO

Datasource

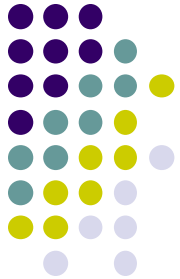
- Creates the helper
- Access to the db
- Opens and closes the connection
- the factories

```
public class LibraryDataSource {

  private final LibraryDBHelper helper;
  private SQLiteDatabase db;

  public LibraryDataSource(Context context) {
    helper = new LibraryDBHelper(context);
  }

  public SQLiteDatabase getDB() {
    if (db == null) open(); // lazy initialization
    return db;
  }

  public void open() throws SQLException {
    db = helper.getWritableDatabase();
  }

  public void close() {
    helper.close();
  }

  // factories
  public BookDAO newBookDAO() {
    return new BookDAO(this); // flyweight ?
  }
}
```

Access to the database

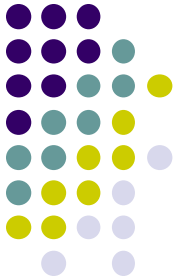
DAO factories

DAO and CRUD



- Reference the datasource
- Create() : creates a record from the POJO, management of its new ID
- Read() : reads a record ID and load according to its fields with the values of the DB
- Update() : updates the database values from the POJO
- Delete() : deletes the record according to the ID POJO

Write data



- A class android.content.ContentValues
- Associative array (Set)
  - Key : name of the base field
  - Value: value of the data in the POJO
- A method put(String key, ???? value) to create the association (overload on each type)
- Methods to retrieve values by key (getAsInteger(), getAsDouble()…)

Create () method of the DAO

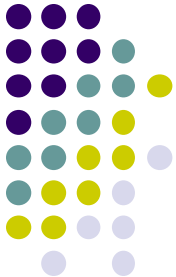
- Create the associative array with the values of the POJO
- Runs the insert query
- Updates the ID
- Returns POJO
- update

```
Public synchronized Book create(Book POJO) {
  // create associative array
  ContentValues values = new ContentValues();
  values.put(COL_TITLE, POJO.getTitle());
  values.put(COL_ISBN, POJO.getIsbn());
  values.put(COL_NBPAGE, POJO.getNbPage());

  // insert query
  int id = (int) getDB().insert(TABLE_NAME, null, values);

  // update id into POJO
  POJO.setId(id);

  return POJO;
}
```

Associative array

API level

update () method of the DAO

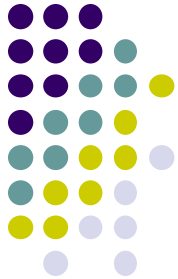


- The same principle as the insert
- The clause also

```
Public synchronized Book update(Book POJO) {
  // create associative array
  ContentValues values = new ContentValues();
  values.put(COL_TITLE, POJO.getTitle());
  values.put(COL_ISBN, POJO.getIsbn());
  values.put(COL_NBPAGE, POJO.getNbPage());

  // where clause
  String clause = COL_ID + " = ?";
  String[] clauseArgs = new String[]{String.valueOf(POJO.getId())};

  // update
  getDB().update(TABLE_NAME, values, clause, clauseArgs);

  // return the POJO
  return POJO;
}
```

Search Expression

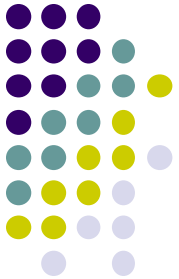
Method delete () DAO



- The where clause, this time!

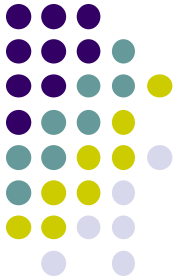
```
Public synchronized void delete(Book POJO) {
  // where clause
  String clause = COL_ID + " = ?";
  String[] clauseArgs = new String[]{String.valueOf(POJO.getId())};

  // delete
  getDB().delete(TABLE_NAME, clause, clauseArgs);
}
```

Read Data



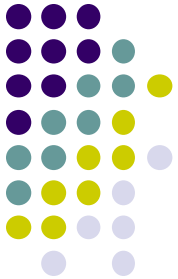
- Two methods :
  - rawQuery (String query, String[] args)
    - Replaces the characters "?" By the argument values passed
    - ex. : rawQuery(“SELECT * FROM book WHERE id = ?”, 1);
  - query (String tableName, String[] columns, String clause, String[] args, String groupBy, String having, String orderBy)
  - Idea, the less possible in the SQL code

Cursor



- Class `android.database.Cursor`

read() Method of DAO



```
public Book read(Book POJO) {
  // columns
  String[] allColumns = new String[]{COL_ID, COL_TITLE, COL_ISBN,   COL_NBPAGE};

  // clause
  String clause = COL_ID + " = ?";
  String[] clauseArgs = new String[]{String.valueOf(POJO.getId())};

  // select query
  Cursor cursor = getDB().query(TABLE_NAME, allColumns, "ID = ?",  clauseArgs, null, null, null);

  // read cursor
  cursor.moveToFirst();
  POJO.setTitle(cursor.getString(1));
  POJO.setIsbn(cursor.getString(2));
  POJO.setNbPage(cursor.getInt(3));
  cursor.close();

  return POJO;
}
```

Search Expression

Request

POJO Chargement

readAll() Method of DAO

```
public List<Book> readAll() {
  // columns
  String[] allColumns = new String[]{COL_ID, COL_TITLE, COL_ISBN, COL_NBPAGE};

  // select query
  Cursor cursor = getDB().query(TABLE_NAME, allColumns, null, null, null, null, null);

  // iterate on cursor and retreive result
  List<Book> books = new ArrayList<Book>();

  cursor.moveToFirst();
  while (!cursor.isAfterLast()) {

    books.add(new Book(cursor.getInt(0),    cursor.getString(1),
                  cursor.getString(2), cursor.getInt(3)));

    cursor.moveToNext();
  }

  cursor.close();

  return books;
}
```

Itération on the recordset

A little refactoring

- A common abstract class for all POJO
- An interface for datasources

```
public abstract class POJO {

  protected int id;

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public POJO(int id) {
    this.id = id;
  }
}
```

```
public interface DataSource {

  SQLiteDatabase getDB();
  void open();
  void close();

}
```

A little refactoring

- An abstract class for all DAO
- Interfaces to add specifics

```
public abstract class DataAccessObject<P extends POJO> {

  private final DataSource datasource;

  public DataAccessObject(DataSource datasource) {
    this.datasource = datasource;
  }

  public SQLiteDatabase getDB() {
    return datasource.getDB();
  }

  // CRUD
  public abstract P create(P POJO);

  public abstract P read(P POJO);

  public abstract P update(P POJO);

  public abstract void delete(P POJO);
}
```

```
public interface AllReadeable<P extends POJO> {

  List<P> readAll();

}
```