

Développement d'applications mobiles

Introduction à la conception d'une application mobile

1. Les systèmes d'exploitation mobiles

Le SE est une plate-forme à bas niveau qui contrôle toutes les fonctionnalités d'un appareil mobile; Il contrôle toutes les opérations de base du téléphone mobile comme option d'écran tactile, cellulaires, Bluetooth, Wifi, appareil photo, lecteur de musique...

2. Les différents systèmes d'exploitation mobiles dominants sur le marché :

Android OS :



Android est un OS gratuit et complètement ouvert (sous licence open source).

C'est une variante de Linux qui fonctionne sur plusieurs appareils tels que les smartphones, tablettes, montres, téléviseurs et voitures.

Il a été développé par Google.

L'App Store officiel des applications Android est le Google Play Store initialement Android Market.

iOS : Iphone OS



Il se trouve non seulement sur les différentes générations de iPhone mais également sur d'autres produits de Apple iPad et iPod touch.

Il est dérivé de Mac OS X dont il partage les fondations : kernel, les services Unix.

La société Apple ne délivre pas de licence du système d'exploitation pour le matériel tiers

L'App Store officiel des applications iOS est : AppStore

Windows Phone:



C'est un système d'exploitation mobile développé par Microsoft pour succéder à Windows Mobile, C'est une évolution de Windows Pocket PC, ancêtre de Windows CE.

Windows Phone est lancé le 21 octobre 2010 en Europe, parmi ses versions :























Windows Phone 7

Windows Phone 7.8

Windows Phone 8 « Apollo ».

L'App Store officiel des applications Windows phone est [Windows store](#).

Les versions de l'Android OS :

>	<input checked="" type="checkbox"/>		Android 8.1.0 (API 27)
>	<input type="checkbox"/>		Android 8.0.0 (API 26)
>	<input type="checkbox"/>		Android 7.1.1 (API 25)
>	<input type="checkbox"/>		Android 7.0 (API 24)
>	<input type="checkbox"/>		Android 6.0 (API 23)
>	<input type="checkbox"/>		Android 5.1.1 (API 22)
>	<input type="checkbox"/>		Android 5.0.1 (API 21)
>	<input type="checkbox"/>		Android 4.4W.2 (API 20)
>	<input type="checkbox"/>		Android 4.4.2 (API 19)
>	<input type="checkbox"/>		Android 4.3.1 (API 18)
>	<input type="checkbox"/>		Android 4.2.2 (API 17)
>	<input type="checkbox"/>		Android 4.1.2 (API 16)
>	<input type="checkbox"/>		Android 4.0.3 (API 15)
>	<input type="checkbox"/>		Android 4.0 (API 14)
>	<input type="checkbox"/>		Android 3.2 (API 13)
>	<input type="checkbox"/>		Android 3.1 (API 12)
>	<input type="checkbox"/>		Android 3.0 (API 11)
>	<input type="checkbox"/>		Android 2.3.3 (API 10)
>	<input type="checkbox"/>		Android 2.3.1 (API 9)
>	<input type="checkbox"/>		Android 2.2 (API 8)
>	<input type="checkbox"/>		Android 2.1 (API 7)
>	<input type="checkbox"/>		Future

Comparaison entre les systèmes d'exploitation

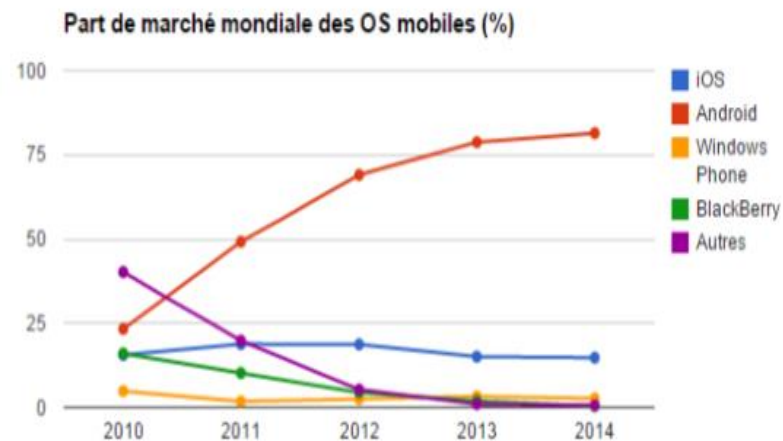
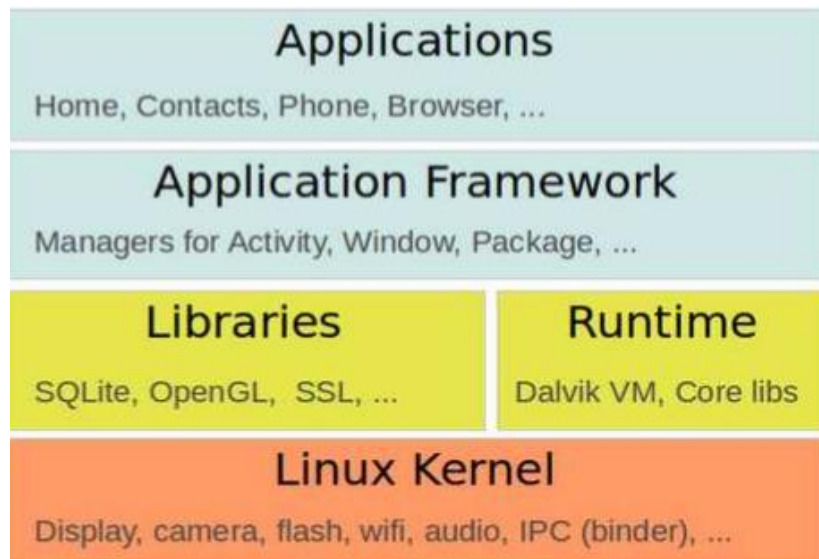


Figure I.6 : Part de marché mondiale des systèmes d'exploitation mobiles [2]

3. Architecture d'un système d'exploitation Android :



1. **Linux Kernel:**

Android s'appuie sur le noyau Linux 2.6 pour les services système de base tels que la sécurité, la gestion de la mémoire et des processus, le réseau et la gestion des drivers. Le noyau sert de couche d'abstraction entre le matériel et le reste de la pile logicielle.

2. **Librairies:**

Des bibliothèques logicielles telles que: WebKit/Blink, OpenGL ES, SQLite ou FreeType ; ces bibliothèques permettent d'exécuter des programmes prévus pour la plate-forme Java ;

3. **Runtime :**

C'est une instance de la machine virtuelle Java, dans laquelle l'application Android s'exécute

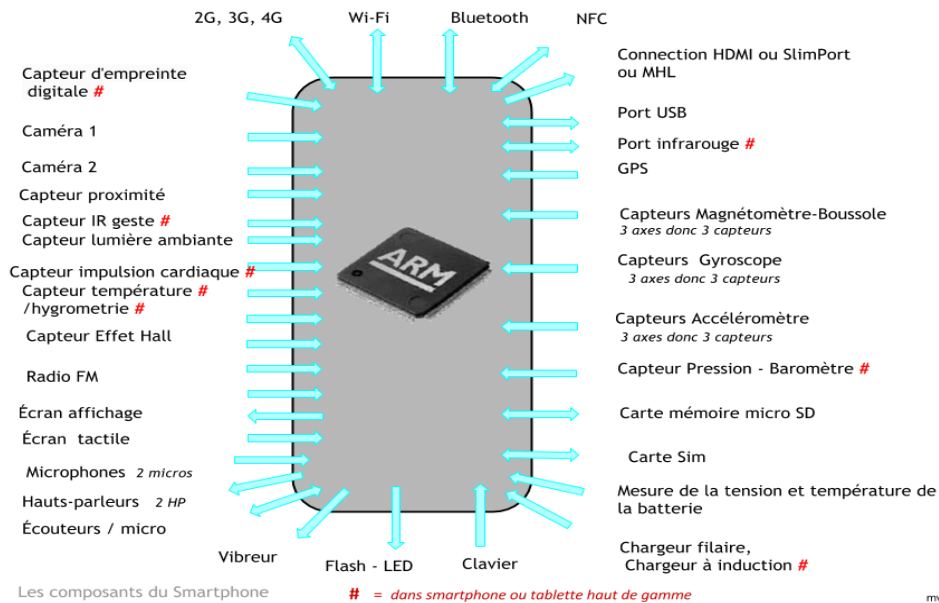
4. **Le Framework d'application :**

C'est un ensemble de services et systèmes sur lesquels se reposent les applications Android pour leurs fonctionnements:

- «Views»
- «Content Providers»
- «Ressource Manager»
- «Activity Manager »

5. Les organes d'un téléphone interagissant avec le microprocesseur.

Un SmartPhone = Un SuperOrdinateur



mv-r3

6. Les parties sensibles du téléphone :



Les applications mobiles :

1. Définition

Une application mobile est un logiciel applicatif développé pour être installé sur un appareil électronique mobile, comme un Smartphone, une tablette ou un baladeur numérique.

Une application mobile peut être soit installée directement sur l'appareil dès sa fabrication en usine, soit téléchargée depuis un magasin d'applications dit «application store» telle que Play Store, l'App Store ou encore le Windows Phone Store ou télécharger à partir d'un serveur à distant.

2. Les types d'application :



a) Applications natives (= Windows Application en VS) :

Les applications natives sont des logiciels conçus spécifiquement pour une plateforme mobile, en utilisant le SDK propre à celle-ci. Les applications ainsi créées sont ensuite téléchargeables depuis une plateforme dédiée au système, généralement un magasin d'application type App Store d'Apple ou google play de Android.

Les langages utilisés sont:

- iOS : ObjectiveC+ Xcode
- Android : Java + Eclipse/ Android Studio/ C#Visual Studio
- Windows : C# + Visual Studio(FrameWork xamarin.Forms)
- Python : Python.

Exemple : Application de Gestion de location de voitures.



b) Applications Hybrides :



Il s'agit d'une application mobile qui fusionne entre les caractéristiques de web application (développement en HTML5 CSS3 et JavaScript) et celles de l'application native. De cette manière, l'application mobile sera accessible sur toutes les plateformes d'application.

Ce type d'application mobile minimise les charges et la durée de son développement même si cela sera au détriment de perfectionnement et de la qualité qui caractérise l'application native.

c) Web Application

Se sont les applications web basées sur les langages (web HTML5 CSS3 et JavaScript....) qui sont lancée par des navigateurs.

3. Déploiement d'une application.

Société	Appareils	Système d'exploitation	Plateforme de téléchargement	Langages de programmation
Apple	iPhone / iPad	iOS	App Store	Objective C / Swift
Google	Samsung / Google Phone et tablette / Motorola / ...	Android	Google Play	Java
Microsoft	Windows Phone	Windows Phone	Windows Store	C#

4. Création d'une application Mobile Native :

a) Préparation de l'environnement :

- 1) Installer Le JDK (Java développement Kit)
- 2) Télécharger et installer Le Genymotion avec Virtual Box (compatibilités).
- 3) Télécharger et Installer Android Studio.

b) Les Emulateurs.

L'Émulateur Genymotion pour Android est une application de bureau qui émule un appareil Android (machine virtuelle). Il fournit un environnement virtualisé dans lequel vous pouvez déboguer et tester des applications Android sans appareil physique.

Exemple 1:

- 1) Démarrer votre Genymotion.
- 2) Installer une version d'Android de votre Emulateur.
- 3) Démarrer l'API de votre choix.

Remarque :

Pour bien exécuter une application Mobile, il faut que sa version doit antérieur à celle de l'émulateur (Prise en charge).

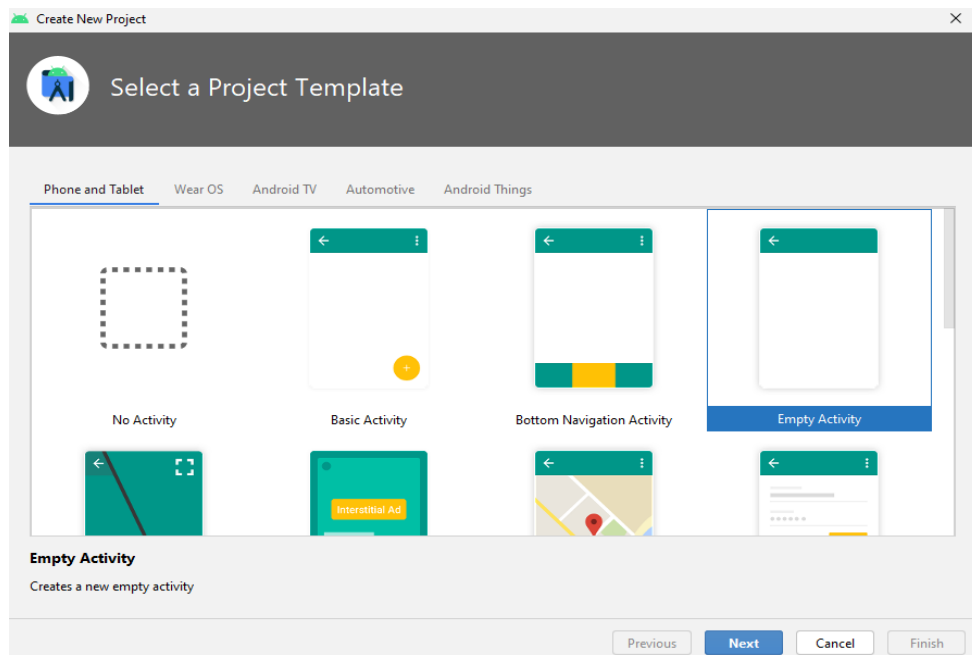
c) Utilisation du devise ou l'appareil physique :

- 1) Télécharger le pilote Mobile de Votre device.
- 2) Brancher votre device et installer son pilote.
- 3) Activer le mode développeur de Votre device.
- 4) Activer le débogage USB de votre téléphone.

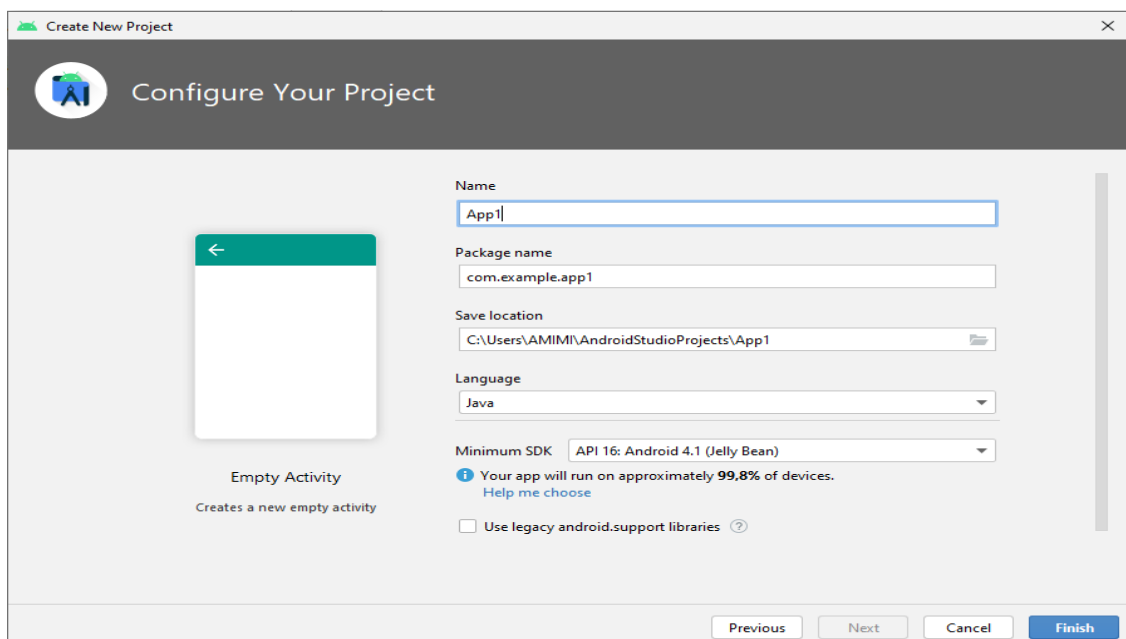
Votre appareil physique sera chargée dans la liste des emulateurs de android studio.

d) Création d'une application native.

- 1) Démarrer votre Emulateur.
- 2) Démarrer Votre android studio et choisir un projet Vide.

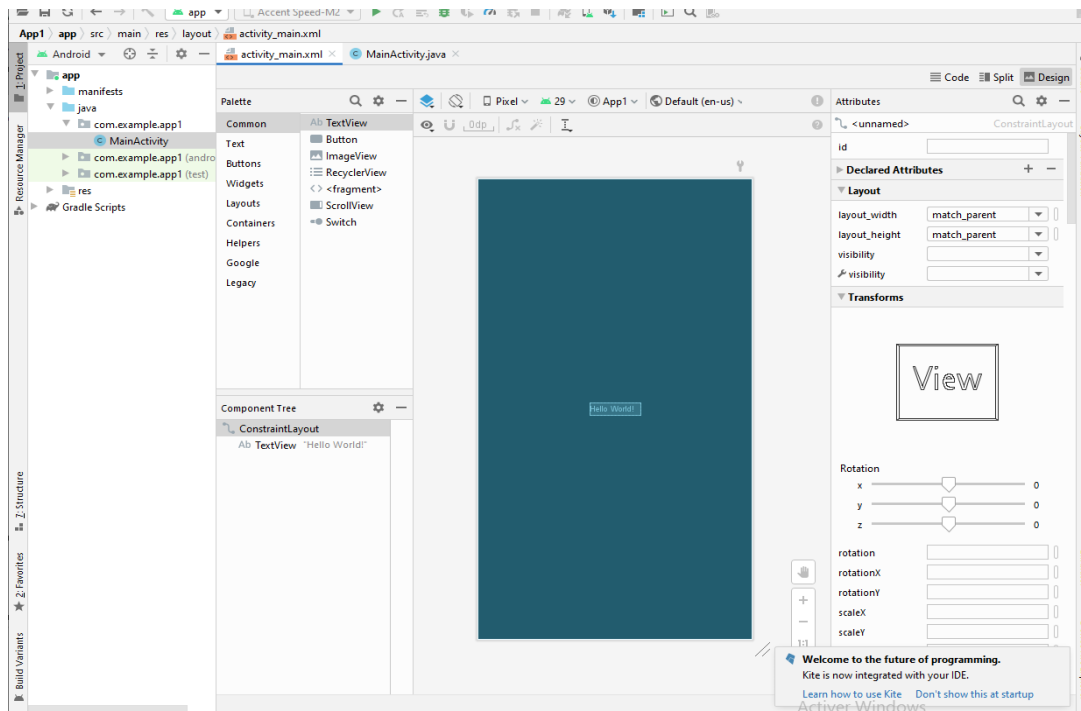


Cliquer sur next et Dans la boîte ouverte saisir ou sélectionner les informations convenables.



- Le nom du projet .
- Le nom du package (unique) utilisé par google Play.
- L'emplacement d'enregistrement du projet.
- Le langage utilisé (ici le JAVA).
- La version de l'API de l'application (**Attention**).

Après le click de fin on aura l'interface suivante :



e) La structure d'un projet sous android studio.

Le volet « project » contient la structure d'une application Native ; elle constituée par :

➤ **Android manifest :**

C'est un fichier xml qui est réservé pour définir les options de votre application Android comme le code de version ou le nom d'affichage de votre application, la liste des permissions requises par votre application, etc.

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

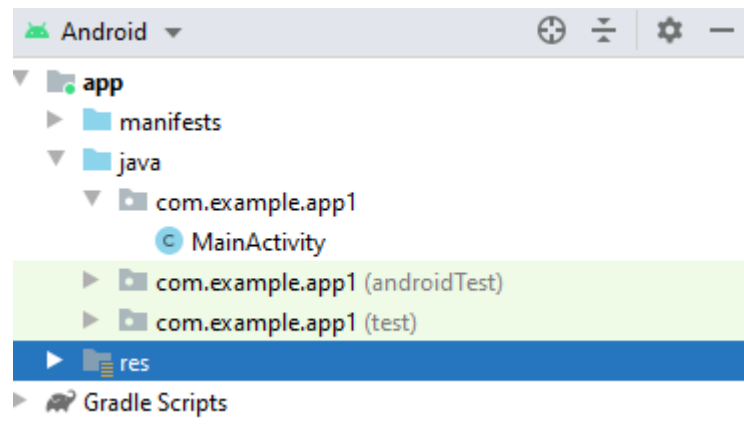
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/Theme.MyApplication">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

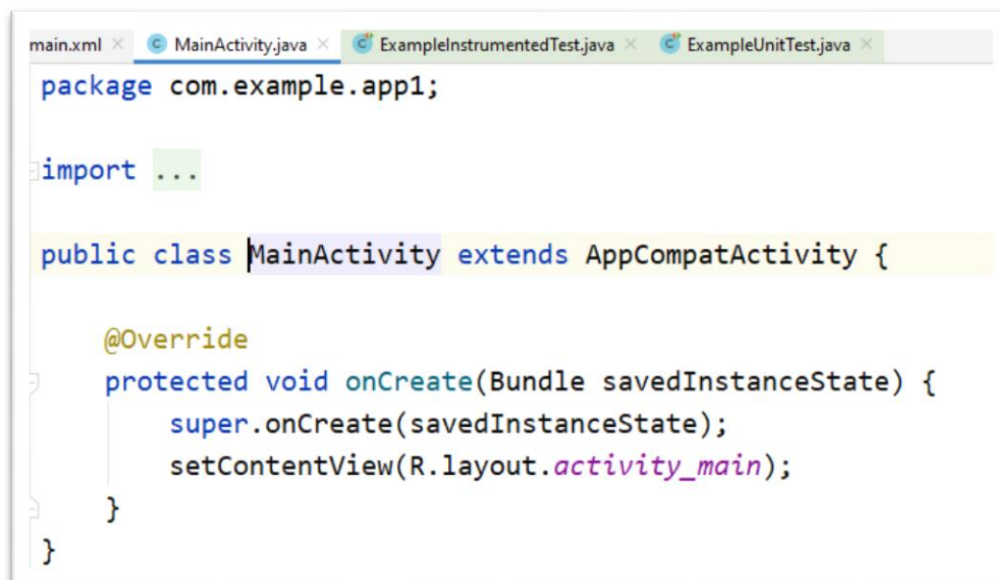
Ce fichier xml définit :

- Les permissions,
- L'icône de l'application,
- Le nom de l'application,
- Les composants (Activité, service, BroadcastReceiver, ContenProvider) de votre application,.....

➤ **Le répertoire Java:**



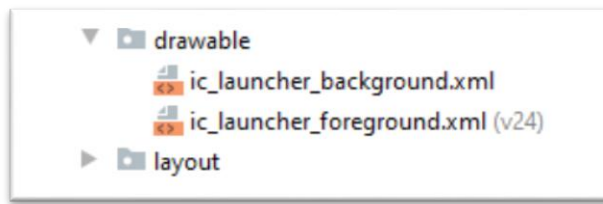
Ce dossier contient les activités créées par le développeur ; se sont tous les fichiers de code source de votre application.



➤ **Le répertoire res :**

C'est le dossier des ressources de l'application ;

- ❖ **Les fichiers drawable** : Ce répertoire contient les images



- ❖ **mipmap**: Ce répertoire contient plusieurs icons utilisées dans l'application tel que l'icon de l'application et résolution différentes.
- ❖ **Layout** : Ce dossier contient les fichiers xml, représentant les interfaces graphique de l'application.
- ❖ **Values** : contient des fichiers sources des valeurs utilisées par l'application tels que :
 - ❖ **colors.xml**: Ce fichier contient les couleurs utilisées dans votre application.
 - ❖ **strings.xml**: Ce fichier contient les chaines de caractères utilisées dans votre application
 - ❖ **themes.xml**: Ce fichier contient tous les themes de l'application

(On va revenir sur ces fichiers).

f) Le fichier activity_main.xml.

Le fichier layout/activity_main.xml est un fichier XML qui contient les balises représentant les composants d'une interface graphique d'une application ; il permet de modifier directement les propriétés ou les attributs des composants de l'interface ; ce fichier est lié à un fichier (*.java) appelé activité (Main_activity.java) qui contient le code suivant :

```
package com.example.app1;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

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

    public void Affiche(View view) {
    }
}
```

Ce fichier contient :

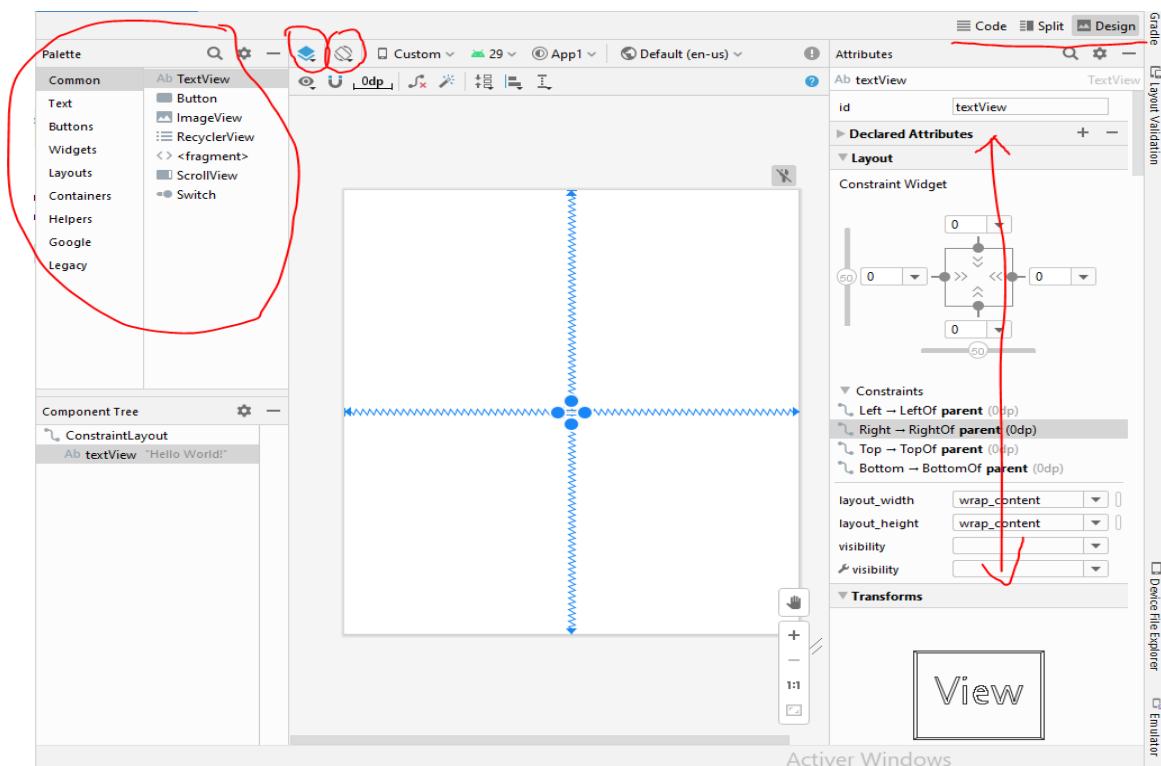
- Le nom du package : **com.example.app1** utiliser dans le Google Play.
- Les bibliothèques implémentées dans le fichier pour utiliser les fonctions et les attributs en relation.
- La classe dérivée **MainActivity** de la classe **AppCompatActivity** ; cette classe contient un classe dérivée **onCreate(Bundle savedInstanceState)** qui permet de charger l'interface en utilisant la méthode

setContentView(R.layout.activity_main);

Contenant le nom du fichier xml à charger (dans notre cas activity_main.xml).

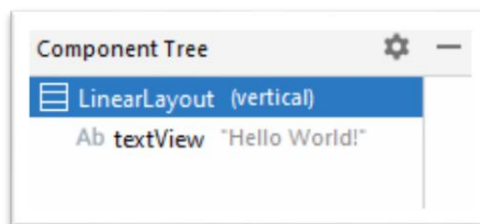
Dans le fichier **MainActivity.java** on ajoute le code source de l'application.

g) Environnement d'android Studio :



h) Structure d'un fichier activity_main.xml (layout):

Changer la racine de ce fichier en utilisant **LinearLayout** (vertical).



Le code xml pour cette layout est :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
```

```

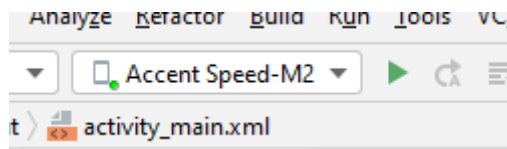
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
tools:context=".MainActivity">

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!" />
</LinearLayout>

```

i) Lancement d'une application :

Sur la barre d'outils ; sélectionner l'émulateur soit sur genymotion ou votre device qui cliquer sur la flèche verte :



5) Les objets de bases d'une application mobile :

a) Définition de Layouts :

Une layout est l'interface graphique représentée par un fichier xml qui est constitué par une Racine et l'empilement des balises représentant les widgets.

b) ConstraintLayout :

C'est un conteneur dans lequel on peut placer des composants (widgets) les un par rapports aux autres (à droite, à gauche, au-dessus , au-dessous).

Parmi les propriétés de cette layout :

- **Layout width** : c'est la largeur du conteneur qui aura les valeurs :

Match_parent : la largeur maximale de son parent.

Wrap_content : Ajustement selon son contenu.

Ou une valeur comme (72dp)

- **Layout height** : c'est la hauteur du conteneur qui aura les valeurs :

Match_parent : la hauteur maximale de son parent.

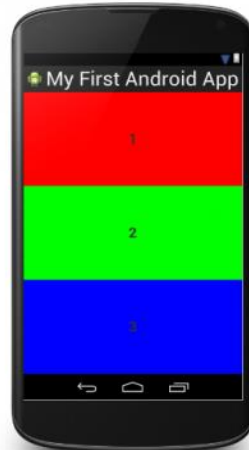
Wrap_content : Ajustement selon son contenu.

Ou une valeur comme (250dp)

c) Les linearLayouts :

C'est un conteneur qui permet de disposer les composants widgets en sens vertical (`android:orientation="vertical"`) ou en sens horizontal (`android:orientation="horizontal"`).

Exemple 1 : `android:orientation="vertical"`)



Exemple 2 : (`android:orientation="horizontal"`)



d) Les widgets :

❖ TextView ou label :

Parmi les propriétés de ce widget :

- `android:id="@+id/textView"`
- `android:layout_width="match_parent"`
- `android:layout_height="72dp"`
- `android:text="Hello World!"`

1. EditText ou Zone de texte :

Parmi les propriétés de ce widget :

- `android:id="@+id/editTextTextPersonName"`

```

➤ android:layout_width="match_parent"
➤ android:layout_height="wrap_content"
➤ android:layout_marginTop="20dp"
➤ android:ems="10"
➤ android:hint="Utilisateur"
➤ android:inputType="textPersonName"
➤ android:textAppearance="@style/TextAppearance.AppCompat.Body2" />

```

Exemple 1: Utilisation de LinearLayout verticale



Le code xml est le suivant :

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="42dp"
        android:layout_marginTop="50dp"
        android:text="Utilisateur"

        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
        android:textColor="#CF3737" />

    <EditText
        android:id="@+id/editTextTextPersonName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:ems="10"
        android:hint="Utilisateur"

```

```

        android:inputType="textPersonName"

        android:textAppearance="@style/TextAppearance.AppCompat.Body2" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="25dp"
            android:text="Mot de passe"

            android:textAppearance="@style/TextAppearance.AppCompat.Display1"
            android:textColor="#C33535" />

        <EditText
            android:id="@+id/editTextTextPassword"
            android:layout_width="match_parent"
            android:layout_height="58dp"
            android:ems="10"
            android:hint="Mot de passe"
            android:inputType="textPassword" />

        <Button
            android:id="@+id/button"
            android:layout_width="match_parent"
            android:layout_height="61dp"
            android:layout_marginTop="25dp"
            android:text="Se connecter"
            android:textAllCaps="false" />

        <Button
            android:id="@+id/button2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="25dp"
            android:text="Rétablir"
            android:textAllCaps="false" />
    </LinearLayout>

```

Exemple 1: Utilisation de LinearLayout horizontale



Le code xml est le suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_marginTop="25dp"
android:orientation="vertical"
tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="63dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center_vertical"
            android:text="Utilisateur:"

            android:textAppearance="@style/TextAppearance.AppCompat.Display1" />

        <EditText
            android:id="@+id/editTextTextPersonName2"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="textPersonName" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="58dp"
        android:layout_marginTop="25dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView3"
            android:layout_width="170dp"
            android:layout_height="match_parent"
            android:layout_gravity="center_vertical"
            android:layout_weight="1"
            android:autoText="false"
            android:freezesText="false"
            android:text="Mot de Passe:"
            android:textAllCaps="false"

            android:textAppearance="@style/TextAppearance.AppCompat.Body2"
            android:textSize="25sp" />

        <EditText
            android:id="@+id/editTextTextPersonName3"
            android:layout_width="wrap_content"
```

```

        android:layout_height="match_parent"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="textPersonName" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:layout_marginTop="25dp"
    android:orientation="horizontal">

    <Button
        android:id="@+id/button3"
        android:layout_width="83dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginLeft="20dp"
        android:layout_weight="1"
        android:text="Se Connector"
        android:textAllCaps="false"
        android:textSize="10sp" />

    <Button
        android:id="@+id/button4"
        android:layout_width="143dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginLeft="20dp"
        android:layout_marginEnd="20dp"
        android:layout_marginRight="20dp"
        android:layout_weight="1"
        android:text="Rétablir"
        android:textAllCaps="false"
        android:textSize="10sp" />
</LinearLayout>
</LinearLayout>

```

2. Button :

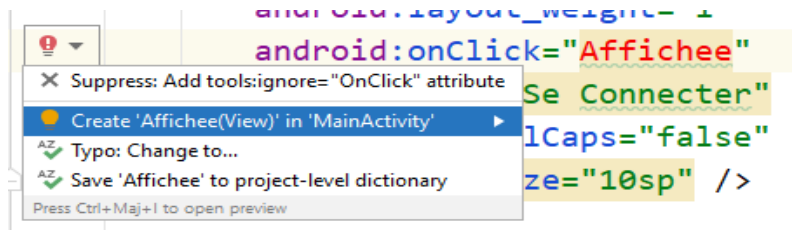
Parmi les propriétés de ce widget :

- `android:id="@+id/button4"`
- `android:layout_width="143dp"`
- `android:layout_height="wrap_content"`
- `android:layout_marginStart="20dp"`
- `android:layout_marginLeft="20dp"`
- `android:layout_marginEnd="20dp"`
- `android:layout_marginRight="20dp"`
- `android:layout_weight="1"`
- `android:text="Rétablir"`
- `android:textAllCaps="false"`
- `android:textSize="10sp"`

5. Associer un événement à un bouton :

Ajouter la ligne : `android:onClick="Affichee"` à la balise Button et générer la méthode Affiche (à gauche de la ligne ci-dessus) cliquer sur create

« Affichee().... » :



La fonction Affichee() s'ajoute au fichier MainActivity.java

```
public void Affiche(View view) {
}
```

Ou bien dans la fenêtre des propriétés rechercher l'évènement onclick et taper votre fonction Affichee ; et ensuite générer la fonction comme ci-dessus.

Sans ajouter un évènement au fichier xml, on peut créer Aussi un écouteur d'évènement dans le fichier MainActivity.java:

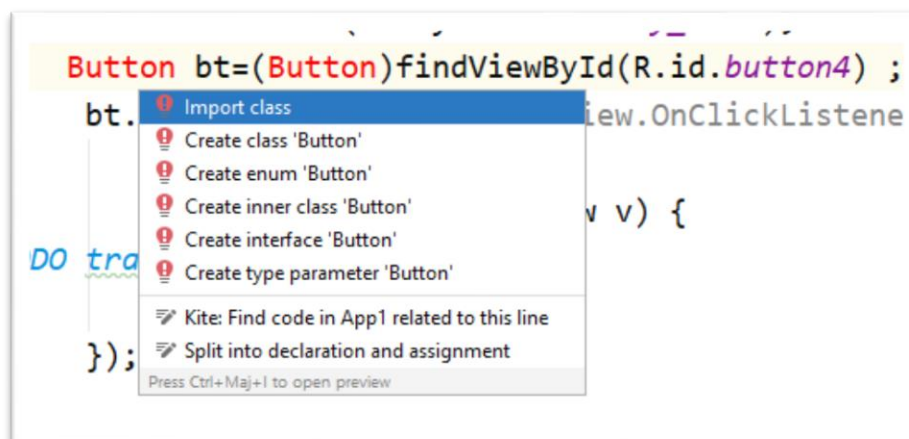
Ajouter la ligne suivante dans la méthode onCreate().

```
Button bt=(Button)findViewById(R.id.button4) ;
```

Les composants généraux de l'interface ne sont pas pris en charge par android studio comme les autres systèmes ; pour cela il faut ajouter la bibliothèque correspondante.



En cliquant sur alt+entree on affiche le menu suivant :



Cliquer sur import class ; la ligne suivant s'ajoute à votre MainActivity.java.

```
import android.widget.Button;
```

En fin on écrit le code suivant :

```
Button bt=(Button)findViewById(R.id.button4) ;
bt.setOnClickListener(new View.OnClickListener() {
```

```
@Override
public void onClick(View v) {
    // Ecrire votre code ici
}
});
```

Exemple 1:

1. Concevoir une interface qui permet de lire votre nom et prénom et afficher votre nom complet.
2. Ecrire le code correspondant.
3. Ajouter le bouton rétablir pour rétablir le contenu de l'interface.
4. Réponse :

Le code XML est le suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="25dp"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="63dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView"
```

```

        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:gravity="center_vertical"
        android:text="Nom"

        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
    />

    <EditText
        android:id="@+id/t1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="text" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="58dp"
    android:layout_marginTop="25dp"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView3"
        android:layout_width="170dp"
        android:layout_height="match_parent"
        android:layout_gravity="center_vertical"
        android:layout_weight="1"
        android:text="Prénom:"
        android:textAllCaps="false"

        android:textAppearance="@style/TextAppearance.AppCompat.Body2"
        android:textSize="25sp" />

    <EditText
        android:id="@+id/t2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="text" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:layout_marginTop="25dp"
    android:orientation="horizontal">

    <Button
        android:id="@+id/bt"
        android:layout_width="83dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginLeft="20dp"

```

```

        android:layout_weight="1"
        android:text="Afficher"
        android:textAllCaps="false"
        android:textSize="10sp" />

<Button
    android:id="@+id/rst"
    android:layout_width="143dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="20dp"
    android:layout_marginLeft="20dp"
    android:layout_marginEnd="20dp"
    android:layout_marginRight="20dp"
    android:layout_weight="1"
    android:text="Rétablir"
    android:textAllCaps="false"
    android:textSize="10sp" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="63dp"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:text="Résultat"

        android:textAppearance="@style/TextAppearance.AppCompat.Display1"
    />
    </LinearLayout>
</LinearLayout>

```

Le code de mainActivity.java est :

```

TextView t1=(TextView)findViewById(R.id.t1);
TextView t2=(TextView)findViewById(R.id.t2);
TextView t3=(TextView)findViewById(R.id.textView4);
Button bt=(Button)findViewById(R.id.bt) ;
Button rst=(Button)findViewById(R.id.rst) ;

bt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String x= t1.getText();
        String y= t2.getText().toString();
        String z=x+" "+y;
        t3.setText(z);
    }
});

```

Exemple 2:

5. Concevoir une interface qui permet de calculer la somme de deux nombres entiers.

6. Ecrire le code correspondant.
 7. Ajouter le bouton rétablir pour rétablir le contenu de l'interface.
- Réponse :



Le code xml est :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_marginTop="25dp"
android:orientation="vertical"
tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="63dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center_vertical"
            android:text="Nombre 1"

            android:textAppearance="@style/TextAppearance.AppCompat.Display1" />

        <EditText
            android:id="@+id/t1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="number" />

    </LinearLayout>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Calculer" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Rétablir" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Résultat" />

</LinearLayout>
```

```

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="58dp"
    android:layout_marginTop="25dp"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView3"
        android:layout_width="170dp"
        android:layout_height="match_parent"
        android:layout_gravity="center_vertical"
        android:layout_weight="1"
        android:autoText="false"
        android:freezesText="false"
        android:text="Nombre 2:"
        android:textAllCaps="false"

        android:textAppearance="@style/TextAppearance.AppCompat.Body2"
        android:textSize="25sp" />

    <EditText
        android:id="@+id/t2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="number" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:layout_marginTop="25dp"
    android:orientation="horizontal">

    <Button
        android:id="@+id/bt"
        android:layout_width="83dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginLeft="20dp"
        android:layout_weight="1"
        android:text="Calculer"
        android:textAllCaps="false"
        android:textSize="10sp" />

    <Button
        android:id="@+id/rst"
        android:layout_width="143dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginLeft="20dp"
        android:layout_marginEnd="20dp"
        android:layout_marginRight="20dp"
        android:layout_weight="1"
        android:text="Rétablir"

```



```

        android:textAllCaps="false"
        android:textSize="10sp" />
</LinearLayout>

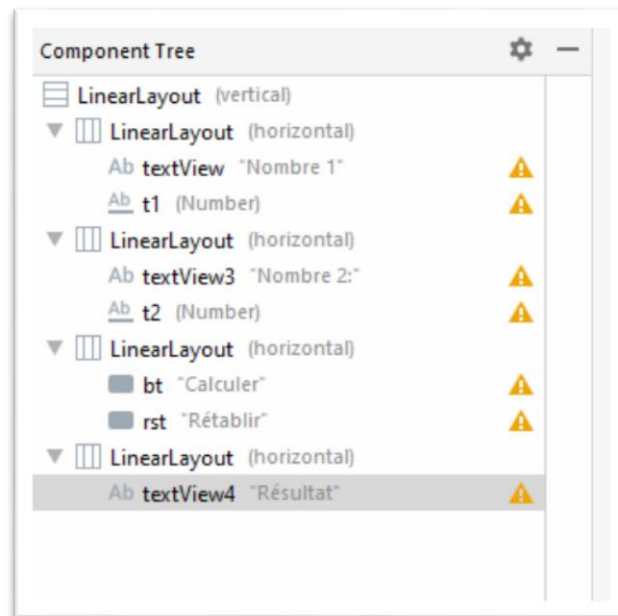
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="63dp"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:text="Résultat"

        android:textAppearance="@style/TextAppearance.AppCompat.Display1" />
    </LinearLayout>
</LinearLayout>

```

La structure du document est :



On passe au code de MainActivity.java.

```

public void onClick(View v) {
    String x= t1.getText().toString();
    String y= t2.getText().toString();
    int z=Integer.parseInt(x)+Integer.parseInt(y);
    t3.setText(Integer.toString(z));
}
ou bien
    bt.setOnClickListener(this);
    public void onClick(View v) {
        t3.setText("autre méthode");
    }

```

Pour rétablir l'interface on utilise :

```

    rst.setOnClickListener(this::afficher);

```

```
private void afficher(View view) {
    t3.setText("");
    t1.setText("");
    t2.setText("");
}
```

Exercice d'application : TP1 exercice 1 et 2.

<https://github.com/amimiabderrahman/XAMARIN-TPS/blob/master/TP1.pdf>

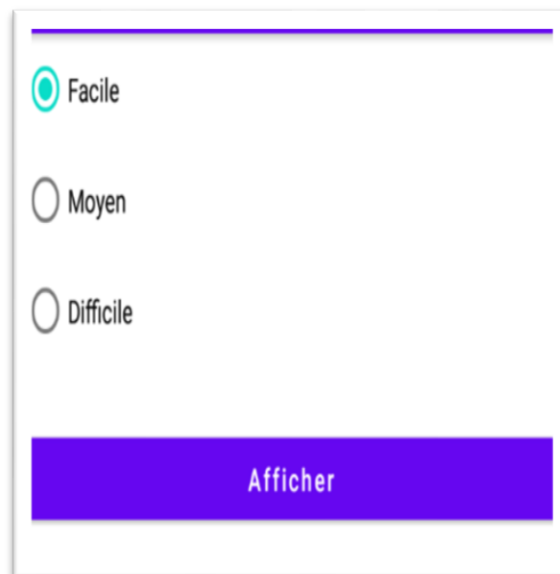
3. RadioButton et les CheckBox :

Parmi les propriétés de ce widget:

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <RadioButton
        android:id="@+id/radioButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="Facile" />
    .....
</RadioGroup>
```

Exemple :



Le code XML est le suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
        <RadioButton
            android:id="@+id/radioButton"
```

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="Facile" />

<RadioButton
    android:id="@+id/radioButton2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Moyen" />

<RadioButton
    android:id="@+id/radioButton3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Difficile" />

<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="25dp"
    android:text="Afficher"
    android:textAllCaps="false" />
</RadioGroup>
</LinearLayout>

```

Pour accéder au composants radioButtons :

```

        bt1=(RadioButton)findViewById(R.id.radioButton);
        bt2=(RadioButton)findViewById(R.id.radioButton);
        bt3=(RadioButton)findViewById(R.id.radioButton);
        rg=(RadioGroup)findViewById(R.id.rg) ;
        t1=(TextView)findViewById(R.id.textView);
        b1=(Button)findViewById(R.id.button);
        b1.setOnClickListener(this::Afficher);

    private void Afficher(View view) {
        //recupérer Le text du radio button coché
        RadioButton rb = (RadioButton)findViewById(rg.getCheckedRadioButtonId());
        t1.setText(rb.getText());
    }

    ou bien

    private void Afficher(View view) {
        //recupérer Le text du radio button coché
        if(bt1.isChecked()) {
            t1.setText("1");
        }
        if(bt2.isChecked()) {
            t1.setText("2");
        }
        if(bt3.isChecked()) {
            t1.setText("3");
        }
    }
}

```

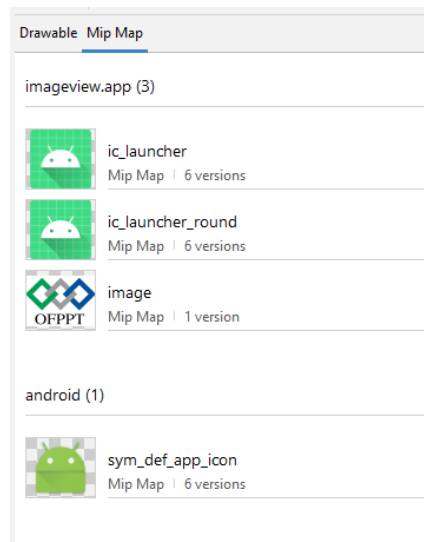
on peut aussi utiliser switch :

```
private void Afficher(View view) {
    switch(rg.getCheckedRadioButtonId())
    {
        case R.id.radioButton:
        {
            t1.setText("facile");
            break;
        }
        case R.id.radioButton2:
        {
            t1.setText("Moyen");
            break;
        }
        case R.id.radioButton3:
        {
            t1.setText("Difficile");
            break;
        }
    }
}
```

TP2 - Ex1 : <https://github.com/amimiabderrahman/XAMARIN-TPS/blob/master/TP2.pdf>

4. ImageView : Pour insérer une image.

En premier lieu glisser votre image vers le dossier mipmap ou drawable ; puis glisser le widget ImageView vers votre layout et choisir l'image soit dans l'onglet drawable ou mipmap.



Puis cliquer sur OK.

Parmi les attributs de ce composant :

```
android:id="@+id/imageView2"
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:srcCompat="@mipmap/image"
```

Le code pour charger une image est :

```
b1=(Button)findViewById(R.id.button);
img=(ImageView)findViewById(R.id.imageView2);
b1.setOnClickListener(this::inserer);
}

private void inserer(View view) {
    this.img.setImageResource(R.drawable.ic_launcher_background);
}
```

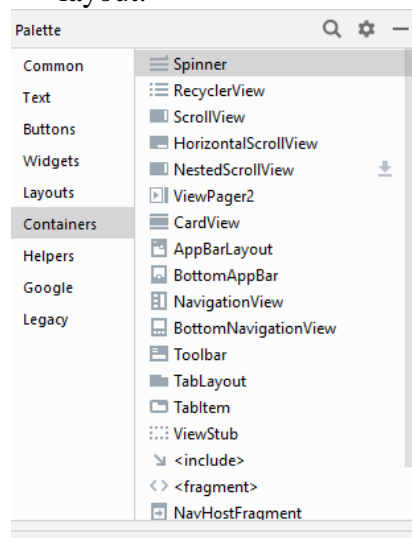
5. ImageButton : Pour insérer un button avec une image.

Parmi les attributs de ce composant :

```
android:id="@+id/simpleImageButtonYouTube"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/simpleImageButtonHome"
android:layout_centerHorizontal="true"
android:layout_marginTop="20dp"
android:background="#005"
android:padding="20dp"
app:srcCompat="@mipmap/image"
```

❖ spinner ou le combobox:

A partir de la palette Containers on glisse le spinner vers notre layout.



Parmi les attributs de ce composant :

```
android:id="@+id/spinner"
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

Android Spinner a 2 modes (modes) avec des interfaces complètement différents:

```
android:spinnerMode="dropdown"
android:spinnerMode="dialog"
```

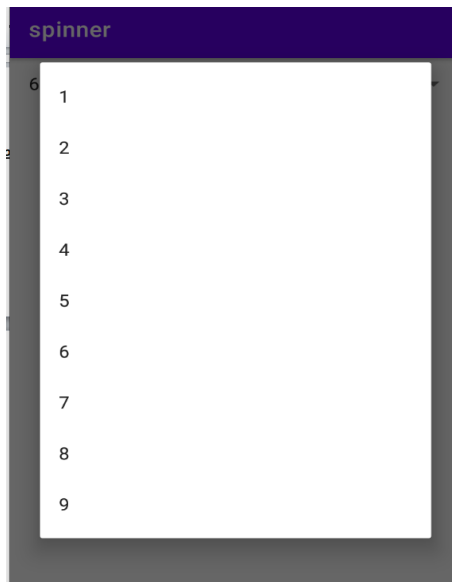
❖ Remplir le spinner :

Exemple 1 : A partir d'un tableau :

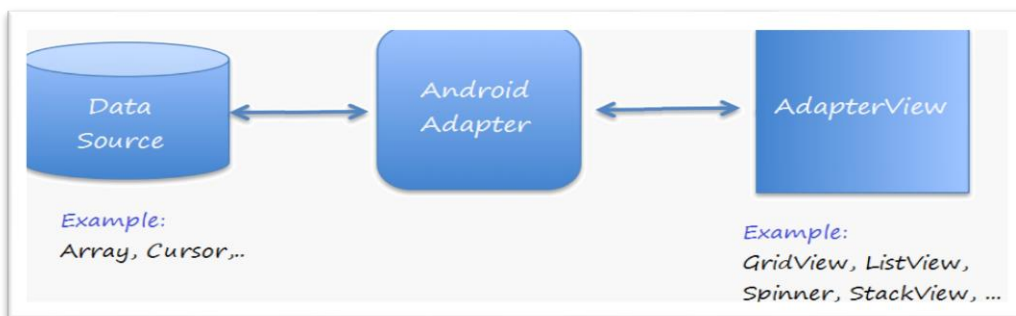
```

sp=(Spinner)findViewById(R.id.spinner);
Integer[]t=new Integer[]{1,2,3,4,5,6,7,8,9};
    ArrayAdapter<Integer> ad
        = new ArrayAdapter<Integer>(this,
android.R.layout.simple_list_item_1 , t);
    sp.setAdapter(ad);
Le Résultat est :

```



Un adapdateur est une classe qui permet de passerdes données d'une base de données (Tableau, liste, arraylist...) vers une interface contenant un spinner.



ArrayAdapter() est un constructeur qui permet de remplir le spinner à partir d'un tableau et choisir le type de Liste pour le spinner.

Exemple 2 : A partir d'un tableau du fichier strings.xml:

Les propriétés du spinner sont :

```

<Spinner
    android:id="@+id/spinner"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
/>

```

Créant un tableau dans le fichier strings.xml
Ouvrir le fichier strings.xml.

```

<string-array name="liste">
    <item>1</item>

```

```

        <item>2</item>
        <item>3</item>
        <item>4</item>
        <item>5</item>
    </string-array>

```

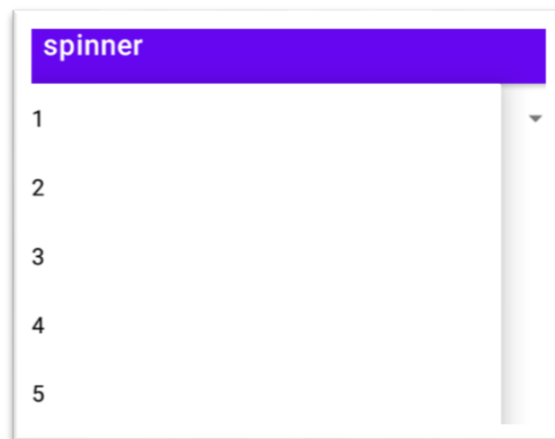
Le code pour remplir le spinner à partir du tableau de strins.xml est :

```

sp=(Spinner)findViewById(R.id.spinner);
ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this,
    R.array.Liste,
    android.R.layout.simple_spinner_dropdown_item);
sp.setAdapter(adapter);

```

Le résultat est :



Pour choisir un élément de spinner ; il faut implémenter des méthodes

```

@Override
public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {

}

@Override
public void onNothingSelected(AdapterView<?> parent) {

}

```

Pour le faire modifier la ligne de la classe MainActivity en ajoutant

```

public class MainActivity extends AppCompatActivity implements
    AdapterView.OnItemClickListener{.....}

```

En suite cliquer sur **implement method** dans le menu contextuel de cette partie soulignée en rouge.

Ajouter la ligne suivante à la méthode `onCreate ()` pour que le spinner ecoute la selection de l'élément.

```

sp.setOnItemClickListener(this);

```

En suite modier la méthode :

```

@Override
public void onItemSelected(AdapterView<?> parent, View view, int position,

```

```
long id) {
String x=parent.getItemAtPosition(position).toString();
t1.setText(x);}

```

Exemple 3 : A partir de ArrayList :

```
sp=(Spinner)(findViewById(R.id.spinner));
liste=new ArrayList();
for(int i=1;i<=10;i++) {

    liste.add(Integer.toString(i));
}
ArrayAdapter ad=new ArrayAdapter(this,
android.R.layout.simple_spinner_dropdown_item,liste);
sp.setAdapter(ad);

```

Exemple 4 : A partir d'une List :

```
Ajouter la bibliothèque : import java.util.List;

sp=(Spinner)(findViewById(R.id.spinner));
liste=new ArrayList();
for(int i=1;i<=10;i++) {

    liste.add(Integer.toString(i));
}
ArrayAdapter ad=new ArrayAdapter(this,
android.R.layout.simple_spinner_dropdown_item,liste);
sp.setAdapter(ad);

```

❖ ListView:

C'est un composant Widget qui permet d'afficher des données empilées dans une colonne.

Parmi ses attributs :

```
<ListView
    android:id="@+id/listView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

```

❖ Remplir une ListView

➤ Parmi les propriétés de ListView :

```
android:choiceMode="singleChoice"

```

ou

```
android:choiceMode="MultipleChoice"

```

➤ A partir d'un tableau :

```
liste1=(ListView)(findViewById(R.id.listView1));
Integer[]t=new Integer[]{1,2,3,4,5,6,7,8,9};
ArrayAdapter ad=new ArrayAdapter(this,
android.R.layout.simple_list_item_1,t);
liste1.setAdapter(ad);

```

➤ A partir de ArrayList:

```
liste1=(ListView)(findViewById(R.id.listView1));
liste=new ArrayList();
for(int i=1;i<=10;i++) {

```



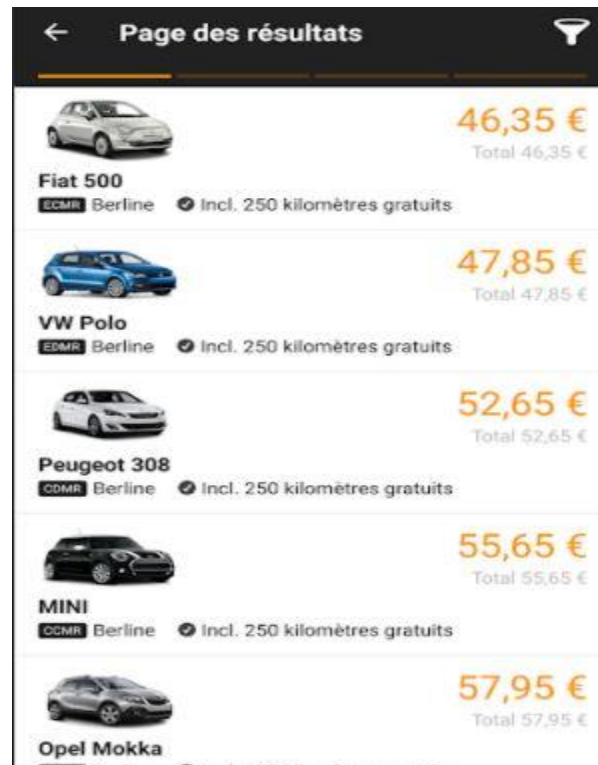
```

        liste.add(Integer.toString(i));
    }
    ArrayAdapter ad=new ArrayAdapter(this,
    android.R.layout.simple_list_item_1,liste);
    liste1.setAdapter(ad);

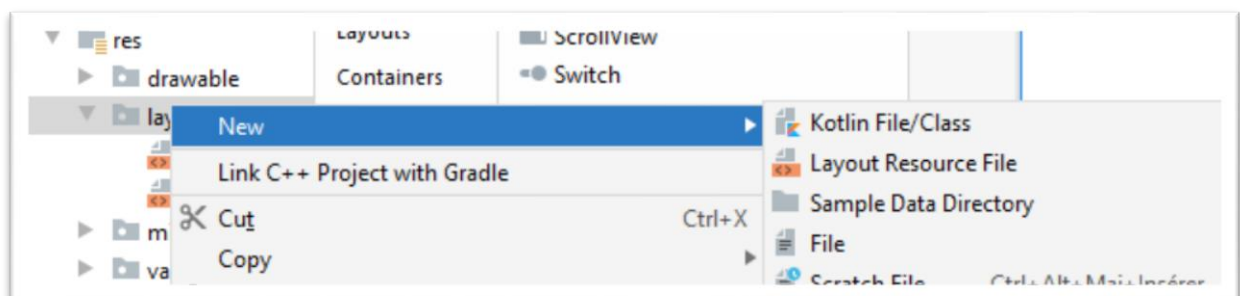
```

➤ Utilisation de la classe BaseAdapter :

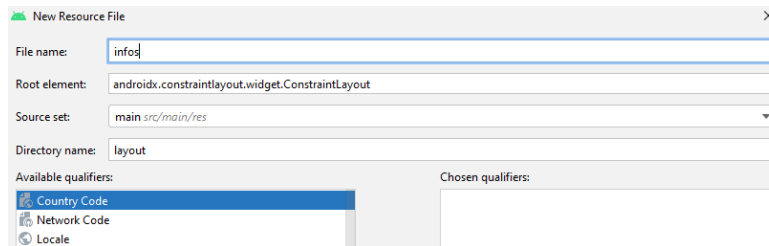
Exemple:



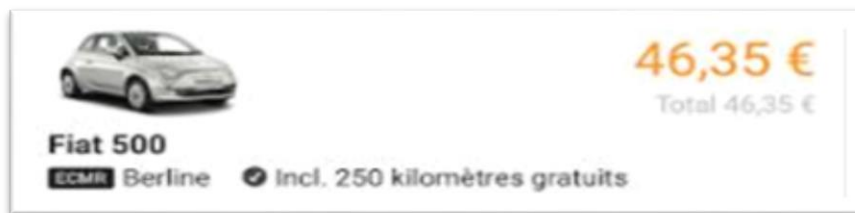
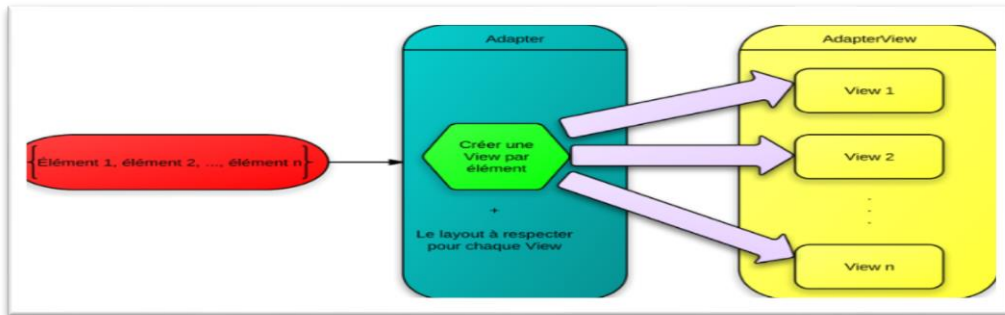
1. Ajouter une Layout(dans le volet à gauche du projet) dans le dossier : res/layout activer le menu contextuel et cliquer sur Layout Ressources File.(infos.xml)



2. La fenêtre suivante s'ouvre :



Taper le nom de la nouvelle layout (Il doit en minuscule)/OK.
Le code xml de cette layout est :



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

    <ImageView
        android:layout_width="181dp"
        android:layout_height="99dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="107dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="49dp"
            android:text="TextView" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="47dp"
```

```

        android:text="TextView" />
    </LinearLayout>
</LinearLayout>

```

3. Créant une classe Voiture (intitule, prix , image) et l'ajouter dans le dossier java/com.exemple/Votreprojet du projet.

```

public class Stagiaire {
    public int image;
    public String nom, prenom;
    public Stagiaire(int image, String nom, String prenom)
    {
        this.image = image;
        this.nom=nom;
        this.prenom=prenom;
    }
}

```

4. Ajouter les images dans Drawable ou mipmap :
Sélectionner les images de votre dossier et les glisser vers Drawable ou mipmap de votre projet.
5. Créer une liste d'objet Voiture et la remplir par des exemples d'enregistrements.

```

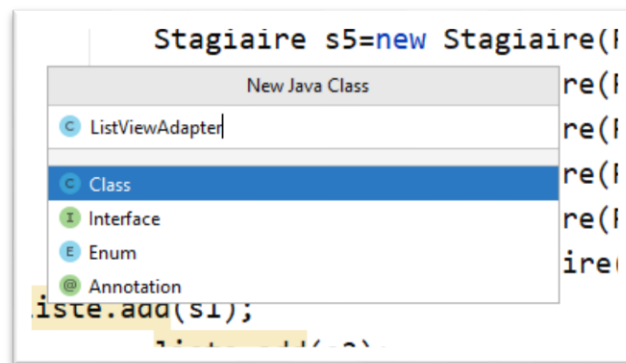
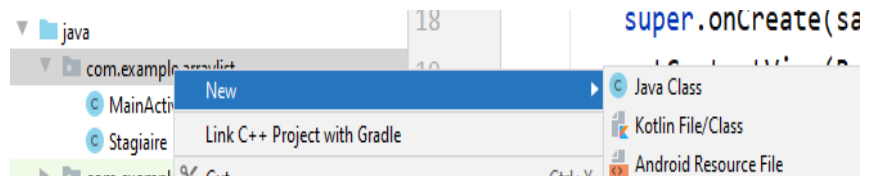
ls=(ListView)(findViewById(R.id.ListView1));
    ArrayList liste=new ArrayList();
    Stagiaire s1=new
Stagiaire(R.drawable.a1,"a1","b");
    Stagiaire s2=new
Stagiaire(R.drawable.a1,"a2","b");
    Stagiaire s3=new
Stagiaire(R.drawable.a1,"a3","b");
    Stagiaire s4=new
Stagiaire(R.drawable.a1,"a4","b");
    Stagiaire s5=new
Stagiaire(R.drawable.a1,"a5","b");
    Stagiaire s6=new
Stagiaire(R.drawable.a1,"a6","b");
    Stagiaire s7=new
Stagiaire(R.drawable.a1,"a7","b");
    Stagiaire s8=new
Stagiaire(R.drawable.a1,"a8","b");
    Stagiaire s9=new
Stagiaire(R.drawable.a1,"a9","b");
    Stagiaire s10=new
Stagiaire(R.drawable.a1,"10a","b");
liste.add(s1);
    liste.add(s2);
    liste.add(s3);
    liste.add(s4);
    liste.add(s5);
    liste.add(s6);
    liste.add(s7);
    liste.add(s8);
    liste.add(s9);
    liste.add(s10);

```

6. Créer la classe ListViewAdapter :

On crée une classe ListViewAdapter qui est dérivée de la classe BaseAdapter et qui permet de transférer une liste de données vers ListView.

- Dans le dossier java/com.exemples/Votreprojet ajouter une classe par menu contextuel et la nommer sous : ListViewAdapter



Puis valider par Entrer.

- Hériter cette classe de la classe BaseAdapter ;

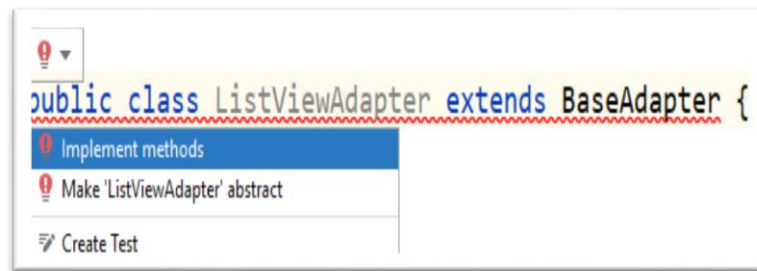
```
public class ListViewAdapter extends BaseAdapter{

}
```

Puis Appuyer sur Alt+Entrer pour charger la bibliothèque :

```
import android.widget.BaseAdapter;
```

Ensuite cliquer sur implement methods :



Le résultat sera :

```
public class ListViewAdapter extends BaseAdapter {
    @Override
    public int getCount() {
        return 0;
    }
    @Override
    public Object getItem(int position) {
        return null;
    }
    @Override
    public long getItemId(int position) {
        return 0;
    }
    @Override
    public View getView(int position, View convertView,
        ViewGroup parent) {
        return null;
    }
}
```

Compléter la classe ListViewAdapter par :

```
package com.example.arraylist;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.constraintlayout.solver.LinearSystem;

import java.util.ArrayList;
import java.util.List;

public class ListViewAdapter extends BaseAdapter {
    private List<Stagiaire> listData;
    private LayoutInflater layoutInflater;
    private Context context;

    public ListViewAdapter(Context aContext, List<Stagiaire>
listData) {
        this.context = aContext;
        this.listData = listData;
        layoutInflater = LayoutInflater.from(aContext);
    }
    @Override
    public int getCount() {
        return listData.size();
    }
}
```

```

    }

    @Override
    public Object getItem(int position) {
        return listData.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup
parent) {

        LayoutInflater=(LayoutInflater)context.getSystemService(Context.LAYOU
T_INFLATER_SERVICE);
        convertView = inflater.inflate(R.layout.infos, parent,
false);
        Stagiaire s = (Stagiaire) this.listData.get(position);
        ImageView img=(ImageView) convertView.findViewById(R.id.image);
        TextView t1=(TextView)convertView.findViewById(R.id.textView);
        TextView
t2=(TextView)convertView.findViewById(R.id.textView2);
        img.setImageResource(s.image);
        t1.setText(s.nom);
        t2.setText(s.prenom);
        return convertView;    }}

```

Récupérer les données d'un item sélectionné :

Dans la méthode onCreate() Ajouter le code de l'événement sur LS :

```

ls.setOnItemClickListener(new AdapterView.OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> a, View v, int
position, long id) {
        Object o = ls.getItemAtPosition(position);
        Stagiaire s = (Stagiaire) o;
        Toast.makeText(MainActivity.this, "Selected :" + " " +
s.nom, Toast.LENGTH_LONG).show();
    }
});

```

➤ A partir des tableaux:

ListviewAdapter :

```

package com.example.arraylist;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

```

```

import androidx.constraintlayout.solver.LinearSystem;

import java.util.ArrayList;
import java.util.List;

public class ListViewAdapter extends BaseAdapter {
    // Ajouter Les attributs en relation avec la Layout Gabarit
    Context context;
    int[] images;
    String[] noms;
    String[] prenom;
    private LayoutInflater inflater;
    // Ajouter Le constructeur
    ListViewAdapter(Context context,
int[] images,String[] noms,String[] prenom)
    {
        super();

        this.context=context;
        this.images=images;
        this.noms=noms;
        this.prenom=prenom;
    }
    @Override
    public int getCount() {
        return noms.length;
    }

    @Override
    public Object getItem(int position) {
        return noms[position];
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        inflater=(LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = inflater.inflate(R.layout.infos, parent,
false);
        ImageView img=(ImageView) convertView.findViewById(R.id.image);
        TextView t1=(TextView)convertView.findViewById(R.id.textView);
        TextView t2=(TextView)convertView.findViewById(R.id.textView2);
        img.setImageResource(images[position]);
        t1.setText(noms[position]);
        t2.setText(prenom[position]);
        return convertView;
    }
}

MainActivity :

ls=(ListView)(findViewById(R.id.listView1));

```

```

int[] images={R.drawable.a1,R.drawable.a2,R.drawable.a3,R.drawable.a4,R.drawable.a5,R.drawable.a6,R.drawable.a7,R.drawable.a8,R.drawable.a9,R.drawable.a10};
String[] noms={"a","a","a","a","a","a","a","a","a","a","a"};
String[] prenomes={"p","p","p","p","p","p","p","p","p","p","p"};
ls.setAdapter(new ListViewAdapter(this, images,noms,prenoms));

Toast.makeText(MainActivity.this, "Selected : " + " " ,
Toast.LENGTH_LONG).show();
ls.setOnItemClickListener(new AdapterView.OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> a, View v, int
position, long id) {
        Object o = ls.getItemAtPosition(position);
        Toast.makeText(MainActivity.this, "Selected : " + " " ,
Toast.LENGTH_LONG).show();
    }
});
}

```

La méthode `LayoutInflater` est une méthode de la classe `LayoutInflater` qui permet de créer une variable sous forme de `View`.

❖ **CalendarView ou DatePicker : Insertion de calendrier.**

Parmi les attributs de ce widget :

```

<CalendarView
    android:id="@+id/calendarView3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

```

On peut la configurer à partir de la fenêtre des propriétés:

```

android:maxDate="31/12/2023"
android:minDate="01/01/2019"
android:firstDayOfWeek="2"

```

Le code pour récupérer la valeur du calendrier est :

```

cv=(CalendarView)findViewById(R.id.calendarView3);
tv=(TextView)findViewById(R.id.textView);
cv.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {
    @Override
    public void onSelectedDayChange(CalendarView view, int year, int
month, int dayOfMonth) {
        // display the selected date by using a toast
        Toast.makeText(getApplicationContext(), dayOfMonth + "/" +
month + "/" + year, Toast.LENGTH_LONG).show();
    }
});

```

❖ **Le scrollView Vertical:**

C'est widget qui permet de définir sont contenu qui dépasse la hauteur du device.

Parmi les propriétés de ce widget :

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">
...
</ScrollView>
```

Dans ce widget scollView , on ajoute une linearLayout Verticale qui contient des boutons comme suit :

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

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

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

            <Button
                android:id="@+id/button30"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button40"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button44"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button45"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button46"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />
```

```
<Button
    android:id="@+id/button31"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />

<Button
    android:id="@+id/button32"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />

<Button
    android:id="@+id/button43"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />

<Button
    android:id="@+id/button42"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />

<Button
    android:id="@+id/button41"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />

<Button
    android:id="@+id/button39"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />

<Button
    android:id="@+id/button38"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />

<Button
    android:id="@+id/button33"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />

<Button
    android:id="@+id/button34"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button" />

<Button
    android:id="@+id/button37"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```

        android:text="Button" />

        <Button
            android:id="@+id/button35"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Button" />

        <Button
            android:id="@+id/button36"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Button" />
    </LinearLayout>
</ScrollView>
</LinearLayout>

```

❖ Le scrollView Horizontal:

Elle permet de défiler les widgets disposés horizontalement.

Exemple :

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <HorizontalScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:orientation="horizontal">

            <Button
                android:id="@+id/button47"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button48"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button49"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button

```

```

        android:id="@+id/button50"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button" />

        <Button
            android:id="@+id/button51"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Button" />

        <Button
            android:id="@+id/button52"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Button" />
    </LinearLayout>
</HorizontalScrollView>
</LinearLayout>

```

❖ Le Switch :

Parmi les propriétés de ce widget sont :

```

<Switch
    android:id="@+id/simpleSwitch"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

```

Le code pour vérifier si elle est active :

```
Boolean switchState = simpleSwitch.isChecked();
```

❖ AnalogClock :

```

<AnalogClock
    android:id="@+id/simpleAnalogClock"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

```

❖ DigitalClock :

```

<DigitalClock
    android:id="@+id/simpleDigitalClock"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:background="#0f0"
    android:padding="30dp"/>

```

Les Boîtes de dialogue

1) La Boîte Toast :

C'est un message transitoire; il ne nécessite aucune intervention de la part de l'utilisateur et ne prend même pas le focus.

```
Toast.makeText(getApplicationContext(), "Message",
Toast.LENGTH_LONG).show();
```

Ou

```
bt=(Button)findViewById(R.id.button53);
bt.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        Toast toast = Toast.makeText(MainActivity.this, "Je suis
un Message!", Toast.LENGTH_SHORT);
        toast.setGravity(Gravity.TOP | Gravity.LEFT, 20, 30);
        toast.show();
    }
});
```

- **duration = Toast.LENGTH_LONG**, signifie que **Toast** s'affichera pendant une longue période, à savoir 3,5 secondes.
- **duration = Toast.LENGTH_SHORT**, signifie que **Toast** s'affichera pendant une courte période, à savoir 2 secondes.

2) La Boîte Toast Personnalisée :

➤ Ajouter une nouvelle layout :

➤ Ajouter des widgets à votre layout :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/toast_layout_root"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#E13B3B"
android:orientation="vertical">

<ImageView
    android:id="@+id/imageView4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:srcCompat="@tools:sample/avatars" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="62dp"
    android:text="Bonjour" />
```

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

```
</LinearLayout>
```

- Ajouter le code suivant à Votre bouton dans

Main_activity :

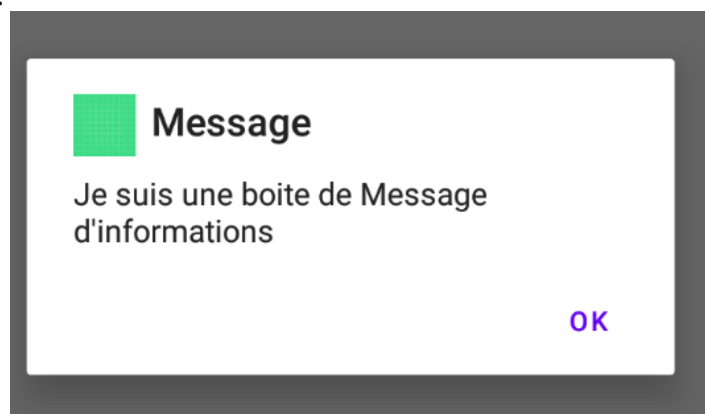
```
Context context = getApplicationContext();
LayoutInflater inflater = getLayoutInflater();
View v0 = inflater.inflate(R.layout.boites, null );
ImageView img=v0.findViewById(R.id.imageView4);
img.setImageResource(R.drawable.ic_launcher_foreground);
Toast toast = new Toast(context);
toast.setView(v0 );
toast.setDuration(Toast.LENGTH_LONG);
toast.show();
```

3) La Boite de Message AlertDialog :

```
AlertDialog.Builder ad=new
AlertDialog.Builder(MainActivity.this);
ad.setTitle("Message");
ad.setMessage("Je suis une boite de Message
d'informations");
ad.setIcon(R.drawable.ic_launcher_background);
ad.setCancelable(false);
ad.setNegativeButton("OK",
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface
dialog, int which) {

Toast.makeText(getApplicationContext(), "Click sur le
boutton OK", Toast.LENGTH_SHORT).show();
        }
    });
```

Le résultat est :



4) La Boite de Confirmation à deux boutons AlertDialog :

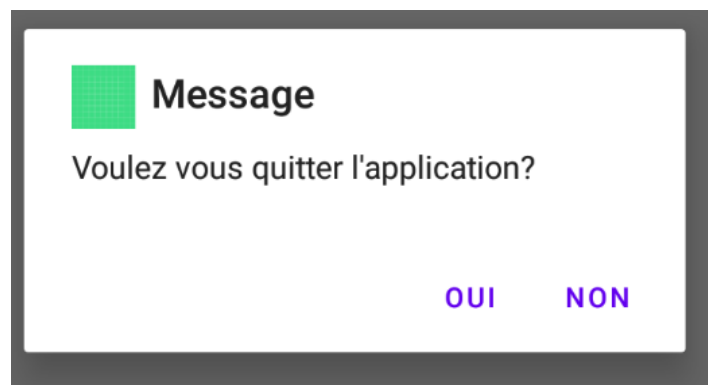
```
AlertDialog.Builder ad=new
AlertDialog.Builder(MainActivity.this);
ad.setTitle("Message");
ad.setMessage("Voulez vous quitter l'application?");
```

```

        ad.setIcon(R.drawable.ic_launcher_background);
        ad.setPositiveButton("Non", new
        DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which)
        {
            Toast.makeText(getApplicationContext(), "Click sur
le bouton OK", Toast.LENGTH_SHORT).show();
        }
    });
    ad.setCancelable(false);
    ad.setNegativeButton("Oui",
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,
int which) {
                Toast.makeText(getApplicationContext(),
"Click sur le bouton Oui", Toast.LENGTH_SHORT).show();
            }
        });
    ad.create();
    ad.show();

```

Le résultat est :



5) La Boite de Confirmation à trois buttons AlertDialog :

Ajouter le code suivant au code ci-dessus :

```

        ad.setNeutralButton("Annuler",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int
which) {
                    Toast.makeText(getApplicationContext(),
"Click sur le bouton Annuler", Toast.LENGTH_SHORT).show();
                }
            });
        ad.create();
        ad.show();

```

6) La Boite de Saisi AlertDialog :

```

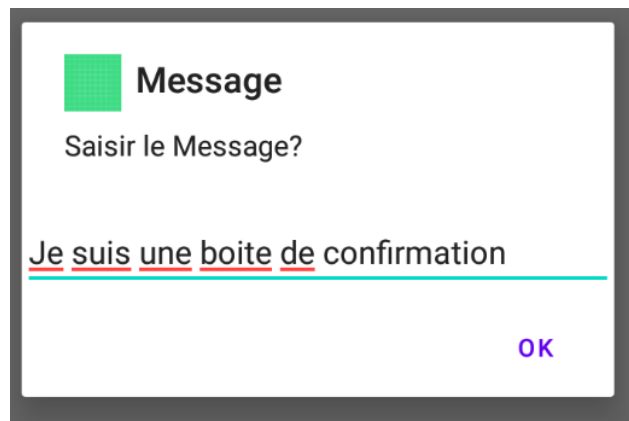
AlertDialog.Builder ad=new
AlertDialog.Builder(MainActivity.this);
ad.setTitle("Message");
ad.setMessage("Saisir le Message?");
ad.setIcon(R.drawable.ic_launcher_background);
EditText ed1=new EditText(MainActivity.this);

```

```
ad.setView(ed1);
```

```
ad.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int
which) {
        Toast.makeText(getApplicationContext(),
"Votre message est : "+ed1.getText(),
Toast.LENGTH_SHORT).show();
    }
});
ad.create();
ad.show();
```

Le résultat est :



7) La Boite personnalisée de AlertDialog :

- Créer une layout de votre choix :

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="55dp"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="29dp"
        android:layout_weight="1"
        android:text="TextView" />

    <EditText
        android:id="@+id/editTextTextPersonName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="72dp"
    android:orientation="horizontal">

    <TextView
```



```

        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="TextView" />

<EditText
    android:id="@+id/editTextTextPersonName2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:ems="10"
    android:inputType="textPersonName"
    android:text="Name" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="65dp"
    android:orientation="horizontal">

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Oui" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Non" />
</LinearLayout>

```

➤ Ajouter le code suivant à Main_activity.java :

```

Button bt=(Button)findViewById(R.id.button);
bt.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        LayoutInflater inflater =
getLayoutInflater();
        View v0 =
inflater.inflate(R.layout.message, null );
        EditText
ed1=v0.findViewById(R.id.editTextTextPersonName);
        EditText
ed2=v0.findViewById(R.id.editTextTextPersonName2);
        Button b1=v0.findViewById(R.id.button2);
        Button b2=v0.findViewById(R.id.button3);
        b1.setOnClickListener(new
View.OnClickListener(){

            @Override
            public void onClick(View v) {

                Toast.makeText(getApplicationContext(), "Bouton 1",

```

```

Toast.LENGTH_LONG).show();
    }
});
AlertDialog.Builder ad=new
AlertDialog.Builder(MainActivity.this);
    ad.setTitle("Message");
    ad.setMessage("Saisir le Message?");

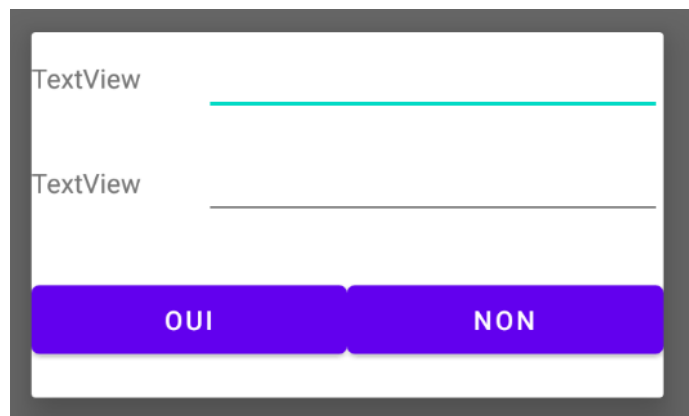
ad.setIcon(R.drawable.ic_launcher_background);
    ad.setView(v0);
    ad.create();
    ad.show();

    }
});
AlertDialog.Builder(MainActivity.this);
    ad.setTitle("Message");
    ad.setMessage("Saisir le Message?");

ad.setIcon(R.drawable.ic_launcher_background);
    ad.setView(v0);
    ad.create();
    ad.show();

```

Le Rsultat est :



Les projets Multi-vues(layouts)

1) Définition :

Les intents sont des messages (intentions ou requetes) qui assurent la communication entres les vues ou layouts d'une application Mobile.

Exemple :

- ❖ Définir un **Intent** qui devrait être utilisée pour lancer une **Activity**.
- ❖ Envoyer des données entre les Vues à travers les intents.
- ❖ Démarrer un service
- ❖

2) Les types d'intents:

- **Intent Explicite** : Ces intents sont généralement utilisés dans une application Mobile ; ils spécifient explicitement le nom du composant cible pour gérer l'intention.

Exemple 1 : Passer des données d'une activité à une autre.

- Créer un nouveau projet sous android studio :



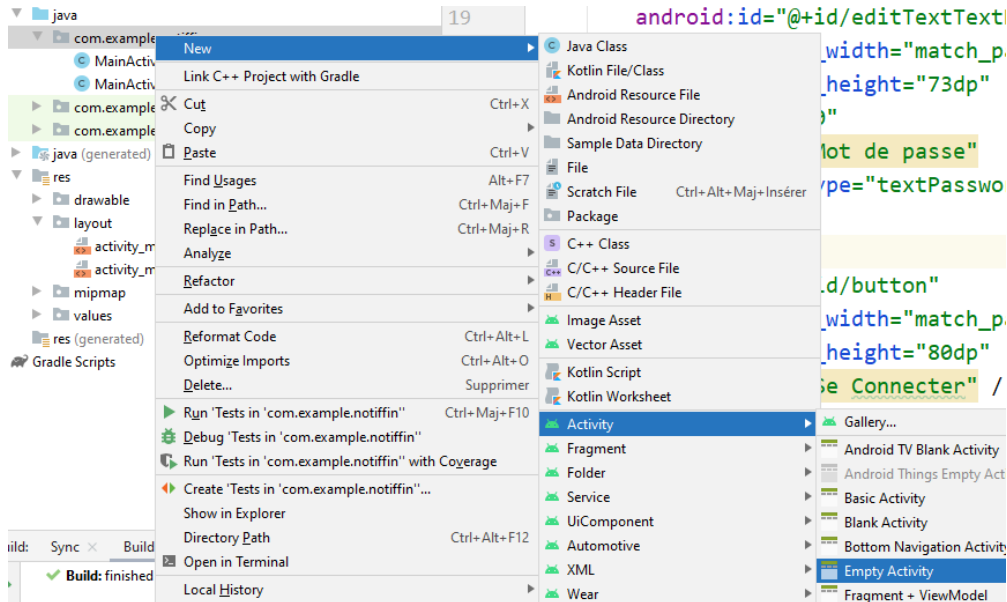
Utiliser le code suivant :

```
<EditText
    android:id="@+id/editTextTextPersonName"
    android:layout_width="match_parent"
    android:layout_height="71dp"
    android:ems="10"
    android:hint="Utilisateur"
    android:inputType="textPersonName" />

<EditText
    android:id="@+id/editTextTextPassword"
    android:layout_width="match_parent"
    android:layout_height="73dp"
    android:ems="10"
    android:hint="Mot de passe"
    android:inputType="textPassword" />
```

```
<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:text="Se connecter" />
```

- Ajouter une nouvelle activité :



- Taper le nom de la nouvelle activité (remarquer l'ajout du fichier xml).

- Remarquer l'ajout d'une ligne dans androidmanifest :
- ```
<activity android:name=".MainActivity3"></activity>
```

- Dans MainActivity Ecrire le code suivant :

```
Button bt=(Button)findViewById(R.id.button);
ed1=(EditText)findViewById(R.id.editTextTextPersonName);
ed2=(EditText)findViewById(R.id.editTextTextPassword);
bt.setOnClickListener(new View.OnClickListener(){
 @Override
 public void onClick(View v) {
 Intent intent=new
 Intent(MainActivity.this,MainActivity2.class);

 intent.putExtra("login",ed1.getText().toString());
 intent.putExtra("pwd",ed2.getText().toString());
 startActivity(intent);

 }
});
```

Ce code permet d'ouvrir une nouvelle activité et transmettre les données de la variable msg1 vers la deuxième activité.

- Dans MainActivity2 Ecrire le code suivant pour l'afficher dans cette activité.

```
TextView tv1=(TextView)findViewById(R.id.textView);
Intent t=getIntent();
String login=t.getStringExtra("login");
String pwd=t.getStringExtra("pwd");
tv1.setText("Votre compte est :"+ login+" "+pwd);
```



Votre compte est :admin 1234

**Exemple 2 :** Passer des données d'une activité à une autre avec Retour .

- Dans ActivityMain ajouter le code xml pour récupérer le résultat retournée:

```
<EditText
 android:id="@+id/res"
 android:layout_width="match_parent"
 android:layout_height="73dp"
 android:ems="10"
 android:hint="Resultat"
 android:inputType="textPersonName" />
```

- Dans MainActivity Ecrire le code suivant :
- ```
Button bt=(Button)findViewById(R.id.button);
ed1=(EditText)findViewById(R.id.editTextTextPersonName);
ed2=(EditText)findViewById(R.id.editTextTextPassword);
bt.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        Intent intent=new
Intent(MainActivity.this,MainActivity2.class);
        intent.putExtra("login",ed1.getText().toString());
        intent.putExtra("pwd",ed2.getText().toString());
        startActivityForResult(intent,0);
    }
});
```

La méthode startActivityForResult() permet de démarrer une activité en fonction du résultat.

admin

...

SE CONNECTER

Resultat

Les notifications

1. Définition :

Une notification est une boîte de dialogue envoyé par un service de l'application. À partir d'API 11, le moteur de création des notifications est la classe Notification builder ; cette classe contient un constructeur qui permet de créer ce moteur de notification.

2. Création du Moteur de notification :

```
Notification notification = new NotificationCompat.Builder(MainActivity.this, "1")
    .setSmallIcon(R.drawable.ic_launcher_foreground)
    .setContentTitle("title")
    .setContentText("message")
    .setPriority(NotificationCompat.PRIORITY_HIGH)
    .setCategory(NotificationCompat.CATEGORY_MESSAGE)
    .build();
```

```
int notificationId = 1;
```

```
notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
```

```
notificationManager.notify(1, notification);
```


Les Menus sous android studio.

Définition :

Contenu

Développement d'applications mobiles	1
Introduction à la conception d'une application mobile	1
1. Les systèmes d'exploitation mobiles	1
2. Les différents systèmes d'exploitation mobiles dominants sur le marché :.....	1
Les versions de l'Android OS :	2
Comparaison entre les systèmes d'exploitation.....	2
3. Architecture d'un système d'exploitation Android :	3
5. Les organes d'un téléphone interagissant avec le microprocesseur.	3
6. Les parties sensibles du téléphone :	4
Les applications mobiles :	5
1. Définition	5
2. Les types d'application :	5
3. Déploiement d'une application.	7
4. Création d'une application Mobile Native :.....	7
5) Les objets de bases d'une application mobile :	13
5. Associer un événement à un bouton :	18