TP-Exercice : Introduction aux Interfaces graphiques utilisateur

Frédéric Guinand / Rodolphe Charrier

frederic.guinand@univ-lehavre.fr

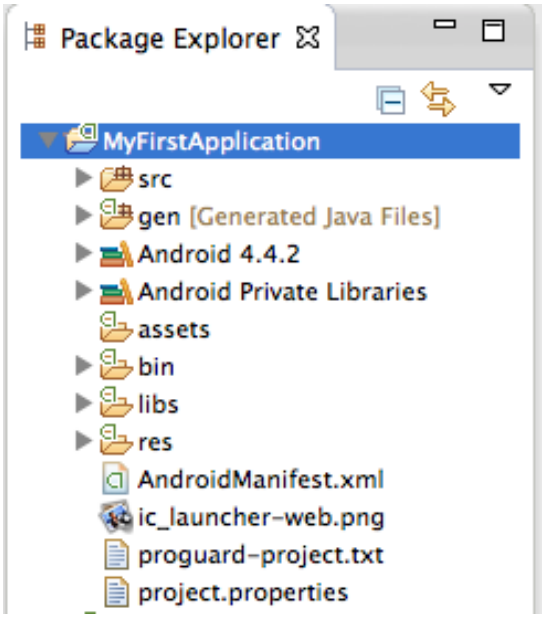
Plan

- Analyse d'une app Android : visite guidée
- Introduction aux interfaces graphiques utilisateur *GUI* (au travers de la création d'une nouvelle application simple) :
  - Layout
  - TextView
  - EditText
  - Button
- Retour sur l'arborescence des fichiers et dossiers d'une app Android.

Première application : visite guidée

Visite guidée

- retour sur l'app "MyFirstApplication" construite par eclipse



Où se situe le code ?

Dans le répertoire
src > mobapp.myfirstapplication

Visite guidée

```java
package mobapp.myfirstapplication;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MyFirstActivity extends Activity {

    protected void onCreate(Bundle savedInstanceState) {

    public boolean onCreateOptionsMenu(Menu menu) {

    public boolean onOptionsItemSelected(MenuItem item) {
}
```

- cela ressemble tout à fait à une application java classique, mais...
- ...
- ...

Visite guidée

```
package mobapp.myfirstapplication;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MyFirstActivity extends Activity {

    protected void onCreate(Bundle savedInstanceState) {

    public boolean onCreateOptionsMenu(Menu menu) {

    public boolean onOptionsItemSelected(MenuItem item) {
}
```

- cela ressemble tout à fait à une application java classique, mais...
- où est le *main* ?
- ...

Visite guidée

```
package mobapp.myfirstapplication;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MyFirstActivity extends Activity {

    protected void onCreate(Bundle savedInstanceState) {

    public boolean onCreateOptionsMenu(Menu menu) {

    public boolean onOptionsItemSelected(MenuItem item) {
}
```

- cela ressemble tout à fait à une application java classique, mais...
- où est le *main* ?
- où sont définis les éléments de l'interface graphique ? Les chaînes de caractères ?

Visite guidée

```
package mobapp.myfirstapplication;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MyFirstActivity extends Activity {

    protected void onCreate(Bundle savedInstanceState) {

    public boolean onCreateOptionsMenu(Menu menu) {

    public boolean onOptionsItemSelected(MenuItem item) {
}
```

- cela ressemble tout à fait à une application java classique, mais...
- où est le *main* ? → **PAS de *main***
- où sont définis les éléments de l'interface graphique ? Les chaînes de caractères ?

Visite guidée

```java
package mobapp.myfirstapplication;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MyFirstActivity extends Activity {

    protected void onCreate(Bundle savedInstanceState) {

    public boolean onCreateOptionsMenu(Menu menu) {

    public boolean onOptionsItemSelected(MenuItem item) {
}
```

- cela ressemble tout à fait à une application java classique, mais...
- où est le *main* ? → PAS de *main*
- où sont définis les éléments de l'interface graphique ? Les chaînes de caractères ? → **ailleurs, pas dans le code java**

Visite guidée

```
public class MyFirstActivity extends Activity {

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

    public boolean onCreateOptionsMenu(Menu menu) {

    public boolean onOptionsItemSelected(MenuItem item) {
}
```

setContentView(R.layout.*activity_my_first*)

$\rightarrow$ R ?

dans le répertoire

gen $>$ mobapp.myfirstapplication $>$ R.java

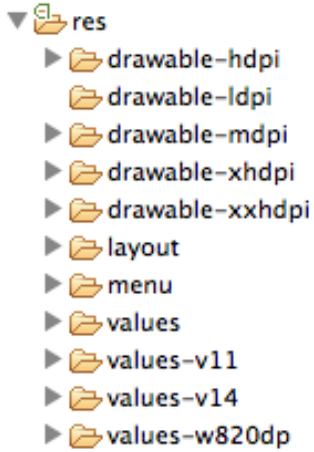
Visite guidée

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package mobapp.myfirstapplication;

public final class R {
    public static final class attr {..}
    public static final class dimen {..}
    public static final class drawable {..}
    public static final class id {..}
    public static final class layout {
        public static final int activity_my_first=0x7f030000;
    }
    public static final class menu {..}
    public static final class string {
        public static final int action_settings=0x7f050002;
        public static final int app_name=0x7f050000;
        public static final int hello_world=0x7f050001;
    }
    public static final class style {..}
}
```

0x7f030000 → static final int : identifiant de ressource unique

Ressources

- Tous les identifiants sont définis dans R.java
- public static final int → accessible depuis **R.type.nom**
- il suffit d'insérer cette référence dans le code java de l'application pour récupérer la ressource



- où sont les ressources ? → dans le répertoire **res/**
- layout > activity_my_first.xml
- values > strings.xml

res/layout/activity_my_first.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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="mobapp.myfirstapplication.MyFirstActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Visite guidée

res/layout/activity_my_first.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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="mobapp.myfirstapplication.MyFirstActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

balises XML (<A>..< /A> or <A.../ >), namespace

res/layout/activity_my_first.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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="mobapp.myfirstapplication.MyFirstActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Attributs

Visite guidée

res/values/strings.xml

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

    <string name="app_name">MyFirstApplication</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>

</resources>
```

@string/hello_world une "sorte" d'indirection
→ pour l'internationalisation

...

Visite guidée

res/values/strings.xml

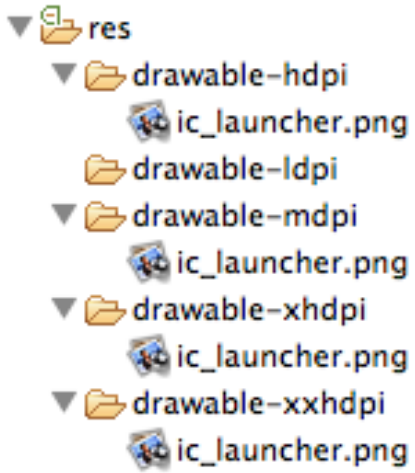
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">MyFirstApplication</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>

</resources>
```

@string/hello_world une "sorte" d'indirection
$\rightarrow$ pour l'internationalisation

où se situe l'icône ?

Visite guidée

il y en a plusieurs versions



- en fonction de la définition de l'écran l'image choisie sera différente
- une définition $\leftrightarrow$ un répertoire *drawable-##dpi* différent

  [souvenez-vous du processus de création de l'application]

En résumé

- un développeur d'application Android devra à minima :
  - écrire quelques lignes de code en java MyApp.java
  - écrire quelques ligne de xml dans plusieurs fichiers pour décrire les ressources de l'application, parmi lequelles : le layout, les chaînes de caractères (éventuellement en différentes langues), les menus, etc.
  - choisir et/ou concevoir d'autres ressources comme les images en plusieurs dimensions pour que l'app soit adaptée à plusieurs types d'écrans.
- à partir de ces fichiers, plusieurs outils de la plateforme Android permettent de produire le package (fichier .apk) qui constituera l'application téléchargeable et installable sur les différents appareils.

Visite guidée

Exécution

- lorsque l'activité (Activity) est lancée, une Interface graphique apparaît à l'écran
  $\Rightarrow$ **comment ?**
- principe de la sérialisation/désérialisation
- sérialiser un objet c'est en proposer une description complète sous la forme d'une chaîne de caractères (équivalent à toString() ou XmlSerializer).
- le désérialiser c'est recréer une nouvelle instance d'objet à partir de sa description sous la forme d'une chaîne de caractères (équivalent à fromString() ou XmlDeserializer).

Visite guidée

Création de l'UI

- sous Android les ressources sont stockées sous la forme de fichier XML
- le fichier xml propose une description de l'objet qui est ensuite recréé au moment de l'appel

setContentView(R.layout.*activity_my_first.xml*)

Visite guidée

Interface graphique

- cette UI est constituée de plusieurs éléments, des Widgets, qui son positionnés les uns par rapport aux autres selon le layout choisi
- layout et widgets sont décrits dans le fichier res/layout/file.xml)
- toutes les chaînes de caractères doivent être définies dans le fichier res/values/strings.xml

Est-ce suffisant pour concevoir une interface graphique simple ?
pas tout à fait mais...
$\rightarrow$ **assez parlé, essayons !!**

Création en pas-à-pas d'une interface utilisateur simple

Multiplication

Objectif

- nous souhaitons créer une application simple pour aider les enfants dans leur apprentissage des tables de multiplication
- l'interface propose une multiplication et demande à l'utilisateur de saisir le résultat

4 x 7 = ?

Eléments nécessaires

- un élément pour afficher l'opération
- un élément pour récupérer la réponse de l'utilisateur
- un élément pour la validation de la réponse

Multiplication

Objectif

- nous souhaitons créer une application simple pour aider les enfants dans leur apprentissage des tables de multiplication
- l'interface propose une multiplication et demande à l'utilisateur de saisir le résultat

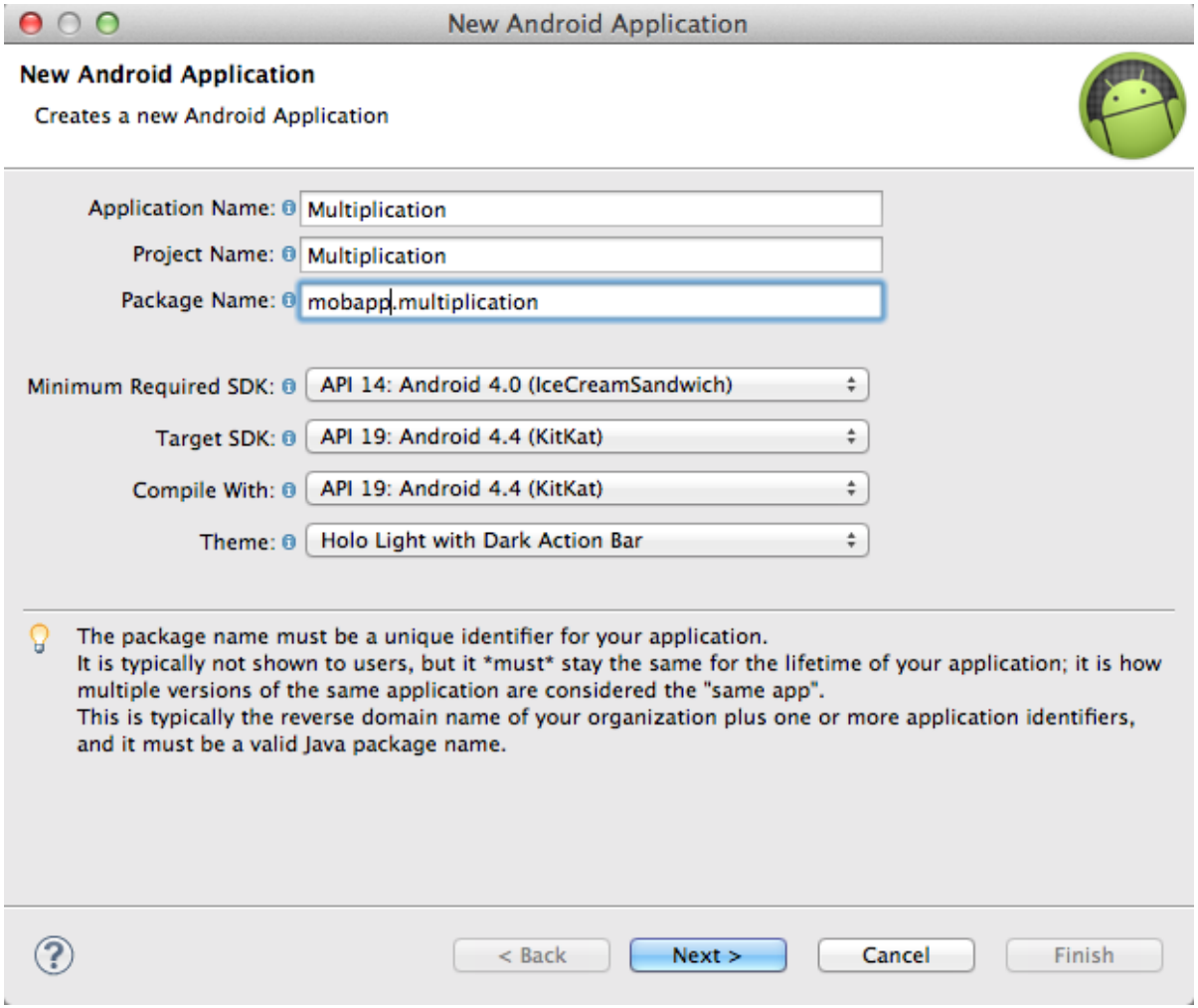
4 x 7 = ?

Eléments nécessaires

- un élément pour afficher l'opération → TextView
- un élément pour récupérer la réponse de l'utilisateur → EditText
- un élément pour la validation de la réponse → Button

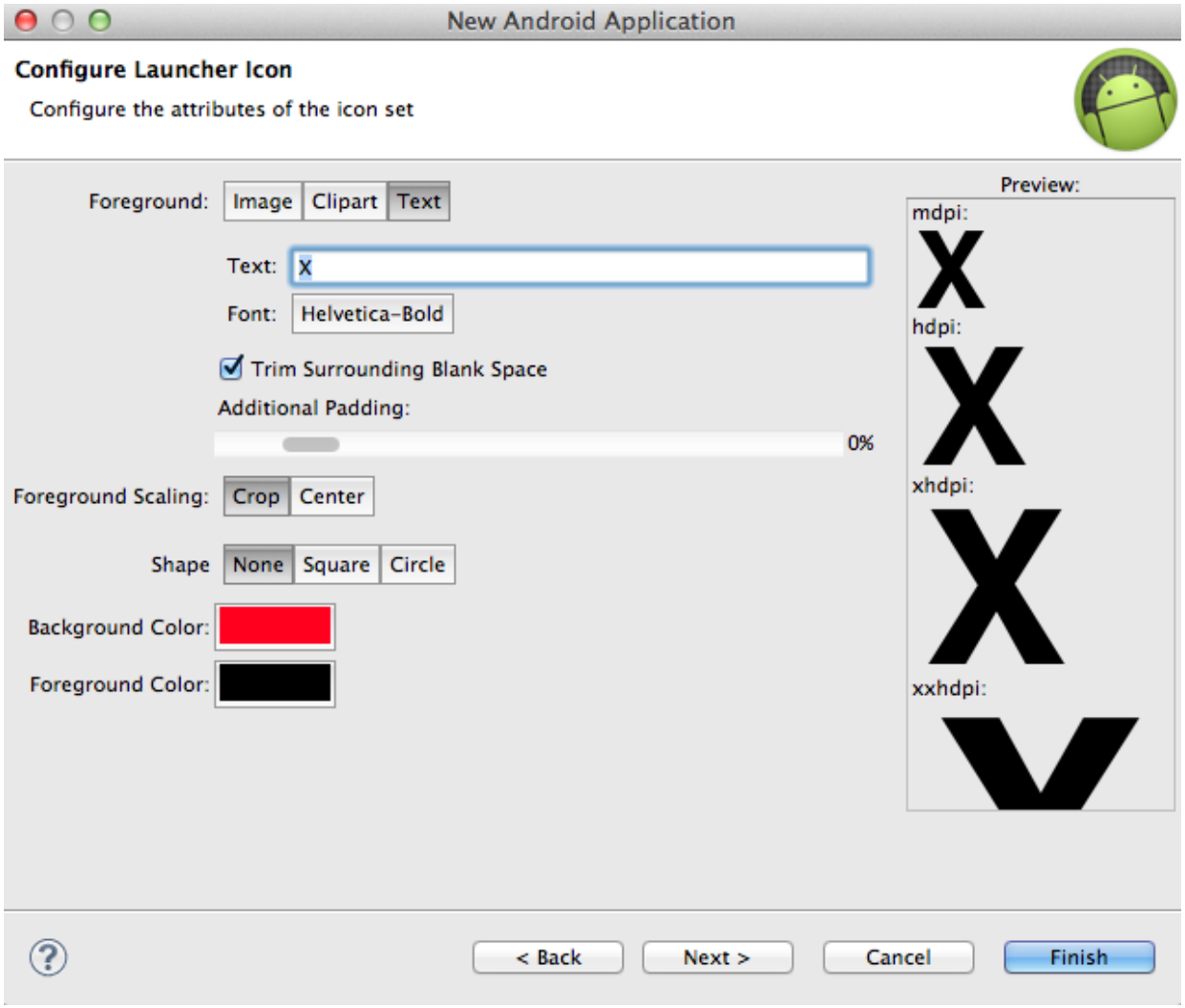
Multiplication - étape 1

Création d'une nouveau projet d'application Android



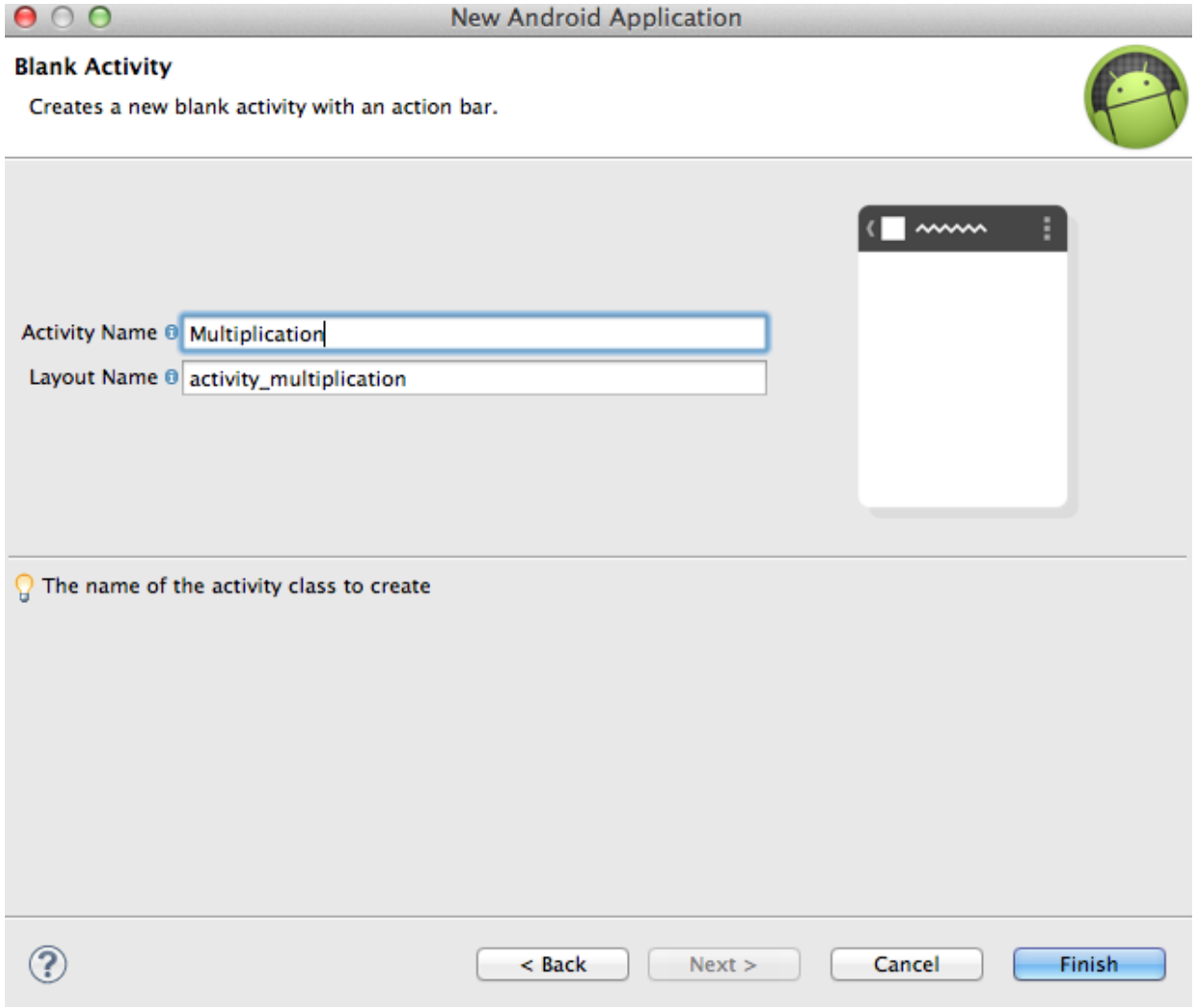
Multiplication - étape 1

Création d'une nouveau projet d'application Android



Multiplication - étape 1

Création d'une nouveau projet d'application Android



Multiplication - étape 2

Modification de l'interface graphique utilisateur

- Modification du layout : RelativeLayout $\rightarrow$ LinearLayout
- Ajout d'un TextView
- Ajour d'un EditText
- Ajour d'un Button

Multiplication - étape 2

Modification du layout : fichier
res/layout/activity_multiplication.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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="mobapp.multiplication.Multiplication" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

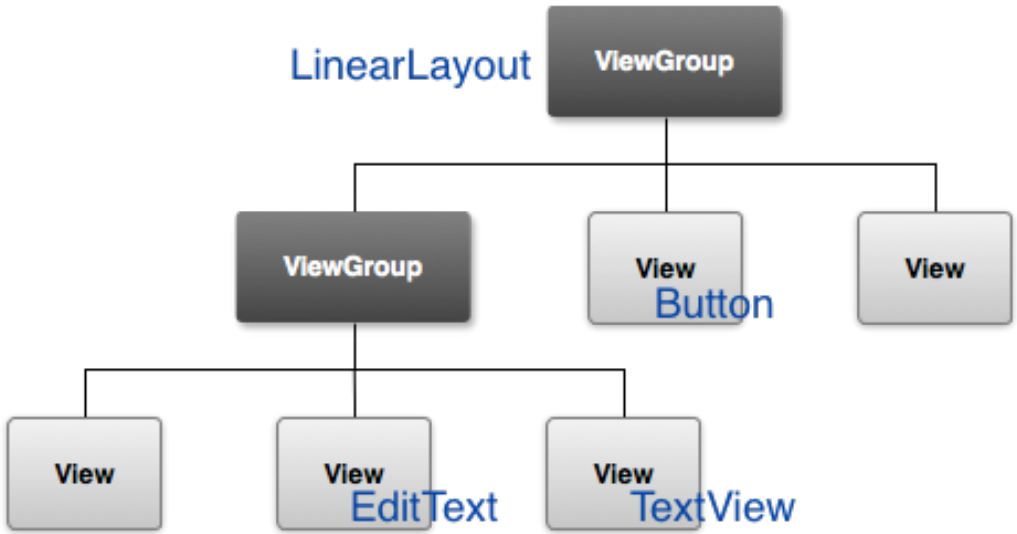
Multiplication - étape 2

Modification du layout : fichier
res/layout/activity_multiplication.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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="mobapp.multiplication.Multiplication" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

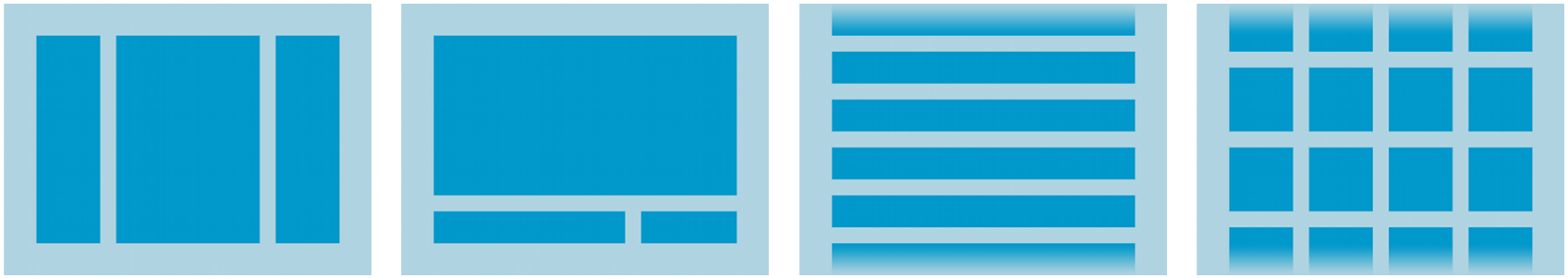
Multiplication - étape 2

- une interface graphique consiste en une hiérarchie de ViewGroups et de Views
- l'utilisateur peut interagir avec les Views (mais pas avec les ViewGroups)
- les Button, TextView, EditText, aussi appélés Widgets, sont des Views



http ://developer.android.com/guide/topics/ui/overview.html

Multiplication - étape 2



Quelques layout

- LinearLayout
- RelativeLayout
- GridLayout

Remarques

→ il est possible d'imbriquer des layout mais attention à la dégradation des performances

→ un layout peut être défini dans un fichier XML ou au moment de l'exécution

Multiplication - étape 2

Modification du layout : fichier
res/layout/activity_multiplication.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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="mobapp.multiplication.Multiplication" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

@string/operation

1. modification RelativeLayout → LinearLayout
2. renommez et changez *hello_world* (qui n'est plus pertinent)

Multiplication - étape 2

Modification du layout

- Modification du fichier res/layout/activity_multiplication.xml
- Ajout de l'attribut : android :orientation="horizontal" (ou vertical)

Effectuez la modification et vérifiez que ça marche
(vous ne devriez pas noter de changement)

Références

- http ://developer.android.com/guide/topics/ui/layout/linear.html (Guide)
- http ://developer.android.com/reference/android/widget/LinearLayout.html (API)

Multiplication - étape 2

Modification du TextView

- android :text="@string/hello_world"

  → android :text="Hello world !" ; aurait été possible également
- cependant dans ce cas la valeur du texte aurait été figée
- @string/hello_world fait référence à une ressource située dans res/values/strings.xml
- les chaînes et les éléments composant le layout doivent être décrits séparément. La raison : l'internationalisation (sera vu ultérieurement)
- Syntaxe : le symbole @ est suivi par le type de l'attribut : @string, @id, etc.

Multiplication - étape 2

Modification du TextView

res/layout/activity_multiplication.xml

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

res/values/strings.xml

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

    <string name="app_name">Multiplication</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>

</resources>
```

Multiplication - étape 2

res/layout/activity_multiplication.xml

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/operation" />
```

res/values/strings.xml

```
<string name="app_name">Multiplication</string>
<string name="operation"> 4 x 7 = </string>
<string name="action_settings">Settings</string>
```

Multiplication - étape 2

res/layout/activity_multiplication.xml

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/operation" />
```

res/values/strings.xml

```
<string name="app_name">Multiplication</string>
<string name="operation">4 x 7 = </string>
<string name="action_settings">Settings</string>
```

Mais...

Multiplication - étape 2

res/layout/activity_multiplication.xml

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/operation" />
```

res/values/strings.xml

```
<string name="app_name">Multiplication</string>
<string name="operation">4 x 7 = </string>
<string name="action_settings">Settings</string>
```

Mais... **comment modifier l'opération lors de la prochaine exécution de l'app ?**

Multiplication - étape 2

Modification du TextView à l'exécution
$\Rightarrow$ modification à la fois de src/Multiplication.java et de res/layout/activity_multiplication.xml

1. un identifiant doit être attribué au TextView : android :id="@+id/operation"

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

le @id/identifier fait référence à un identifiant de ressource, et dans @+id/ le "+" indique que l'identifiant de ressource est défini pour la première fois

Multiplication - étape 2

Modification du TextView à l'exécution
$\Rightarrow$ modification à la fois de src/Multiplication.java et de res/layout/activity_multiplication.xml

1. l'ID du TextView est utilisé dans le code en utilisant **findViewById()** (attention à la casse des caractères)
   TextView theOperation = (TextView) findViewById(R.id.operation) ;
2. ensuite la valeur du TextView peut être changée
   theOperation.setText(the-new-string) ;

Remark

vous pouvez placer ce code à la fin de la méthode onCreate()

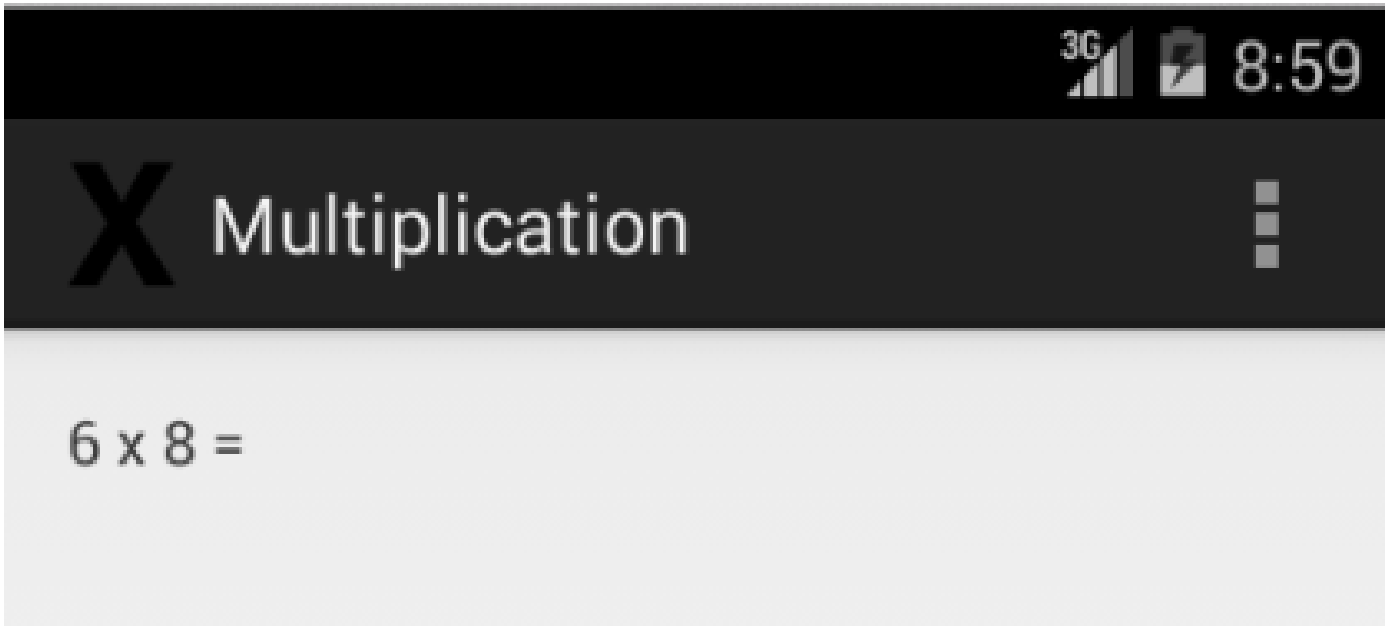
Multiplication - étape 2

Exercice 1

1. Modifiez le code de Multiplication.java en remplaçant à l'exécution l'opération "4 x 7 =" par une autre opération constante ("6 x 8 =" par exemple).
2. Modifiez une nouvelle fois le code pour que :

   à chaque nouvelle exécution de l'app, une nouvelle opération de multiplication (dont les opérandes sont choisis aléatoirement entre 1 et 9) soit proposée à l'utilisateur

Multiplication - étape 2



la suite...

- ajout d'un EditText
- ajout d'un Button

Multiplication - étape 2

Ajout d'un EditText

res/layout/activity_multiplication.xml

```
<EditText
    android:id="@+id/answer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="@string/default_answer" />
```

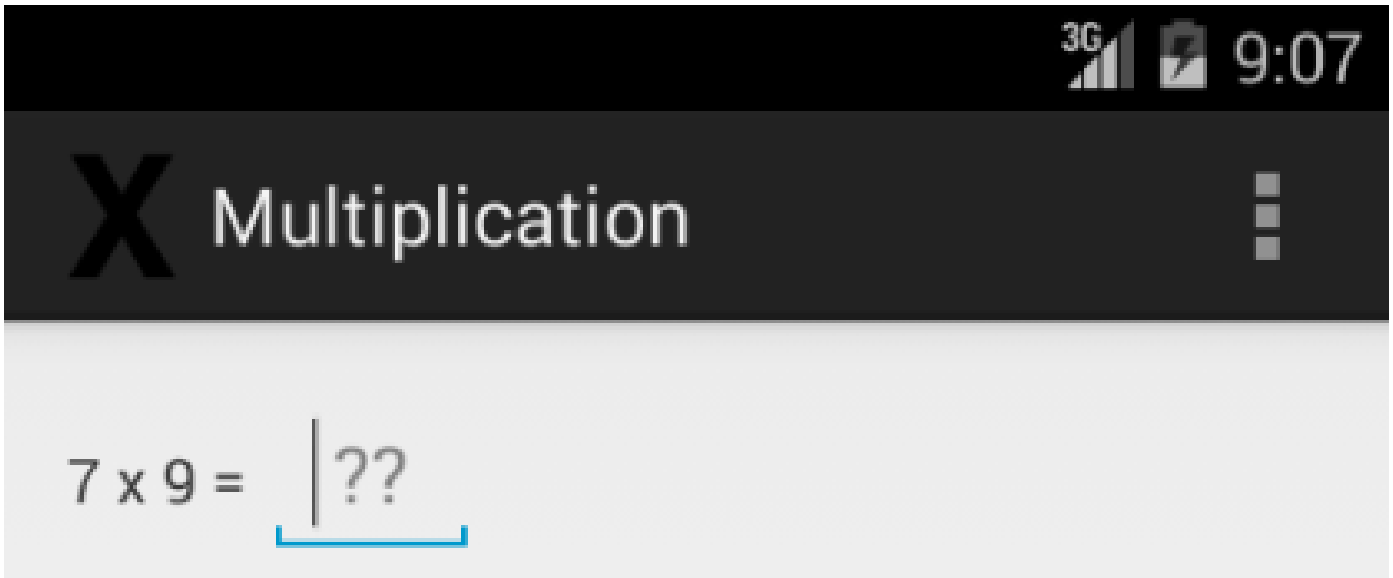
- android :**hint**="@string/default_answer" affiche une chaîne par défaut qui sera recouverte par la réponse saisie par l'utilisateur

Multiplication - étape 2

Ajout d'un EditText

res/values/strings.xml

```
<string name="app_name">Multiplication</string>
<string name="operation"> 4 x 7 = </string>
<string name="action_settings">Settings</string>
<string name="default_answer">" ?? "</string>
```



Multiplication - étape 2

Ajout d'un Button

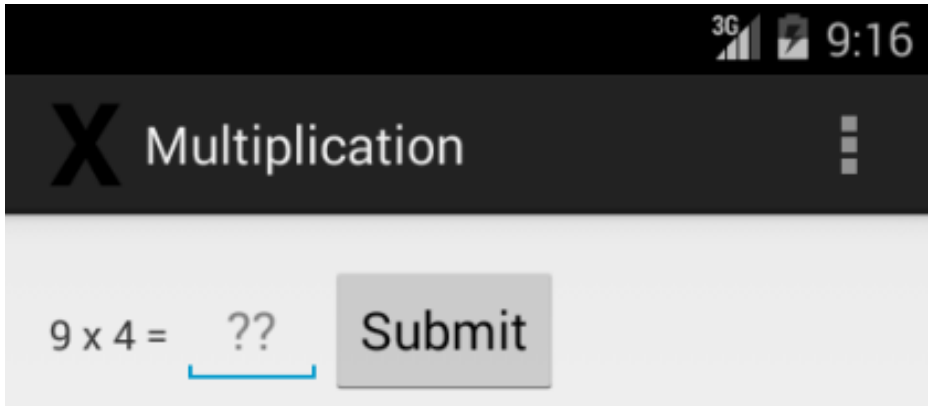
res/layout/activity_multiplication.xml

```
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/submit" />
```

res/values/strings.xml

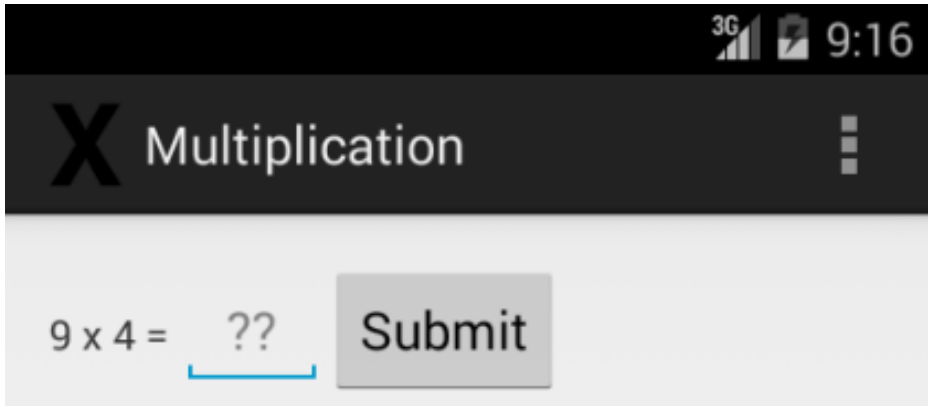
```
<string name="app_name">Multiplication</string>
<string name="operation"> 4 x 7 = </string>
<string name="action_settings">Settings</string>
<string name="default_answer">" ?? "</string>
<string name="submit">Submit</string>
```

Multiplication - étape 2



l'interface semble terminée
mais...

Multiplication - étape 2



l'interface semble terminée
mais...

Que fait le programme lorsque l'utilisateur clique sur le Button ?

Multiplication - étape 3

Comportement de l'app

lorsque l'utlisateur clique sur le bouton l'app doit :

1. exécuter une méthode qui...
2. récupère le contenu de l'EditText
3. vérifie que la réponse donnée est la bonne
4. écrit un message (quelque part) pour informer l'utilisateur de sa réponse (bonne/mauvaise)

mais pour cela lorsque le bouton est pressé cela doit déclencher l'appel à une méthode... **Comment et où ?**

Multiplication - étape 3

res/layout/activity_multiplication.xml

android :onClick="methodName" où methodName est une méthode dans le code java de l'app

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/submit"
    android:onClick="verifyOperation" />
```

src/mobapp.multiplication/Multiplication.java

ajout d'une méthode public void avec un unique paramètre View

```
public void verifyOperation(View view) {

}
```

Exercice

- informez l'utilisateur de la validité de sa réponse :
  - ajoutez unTextView pour écrire cette information
  - modifiez strings.xml et activity_multiplication.xml
- developpez le code pour public void verifyOperation(View view)
  - récupérez le contenu du EditText
  - effectuez l'opération et comparez le résultat avec la réponse donnée par l'utilisateur
  - mettez à jour leTextView qui informe l'utilisateur

un peu plus loin

- ajoutez un nouveau bouton pour quitter l'application
- ajoutez un nouveau bouton qui permette à l'utilisateur de demander une nouvelle multiplication

Pour les méthodes des Widget voir developer.android.com

Retour à la visite guidée

Visite guidée

Répertoires

- src → les sources java du développeur
- gen → les sources java générées par les outils de Android/eclipse
- Android 4.4.2
- Android Private Librairies
- assets
- bin
- libs
- res → ressources (layout, strings, style, images, etc.)

Visite guidée

répertoires virtuels : librairies utilisées par le projet

- "Android 4.4.2" → android.jar : le SDK cible
- "Android Private Librairies" → librairies pour la rétro-compatibilité. En cas de problème de compatibilité vous devez vous intéresser à cette librairie pour trouver une solution (http ://developer.android.com/tools/support-library/index.html).
- vous pouvez également avoir un répertoire "Android Dependencies" → il contient les fichiers jar dont le projet dépend.

ne les supprimez pas !! par contre vous pouvez les ignorer (dans le cadre de ce cours tout au moins)

les ressources

Il y a deux répertoires pour les ressources :

- assets → vous pouvez organiser ce répertoire comme vous le souhaitez, avec des sous-répertoires par exemple. Vous pouvez y placer des données brutes utilisées par votre app.
- res → dans Android, les ressources sont séparées du code. Les éléments de texte (titre de fenêtre, textes des boutons, etc.) ou les images doivent être placés dans ce répertoire. Il est ainsi possible de changer ces données sans changer le code lui-même.

→ Plus d"informations ? Allez lire cet article

http ://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources

répertoire bin

- le bytecode classe.dex
- le package de l'application MyFirstApplication.apk
- beaucoup d'autres éléments nécessaires à la construction de l'apk, dont certains sont copiés depuis d'autres répertoires (images, AndroidManifest.xml).

Visite guidée

les autres fichiers

- ic_launcher-web.png → icône du projet
- proguard-project.txt → pour l'obfuscation de code (inutile pour l'instant - mais par curiosité, ouvrez-le)
- project.properties → précise la version cible de l'API (jetez un oeil à ce fichier)
- **AndroidManifest.xml** ⟶ nous verrons cela plus tard

des questions ?

j'en ai quelques unes pour vous...

1. quels sont les trois fichiers essentiels qu'un développeur doit modifier pour une la création d'une application simple ?
2. quelle est la différence entre un ViewGroup et un View ?
3. quel est le rôle de "@+id/" dans l'attribut d'un widget ? android :id="@+id/truc"
4. si vous ajouter un bouton à votre interface et que vous le déclarez dans le fichier layout.xml file, que devez vous faire pour rendre ce bouton complétement fonctionnel ? plus précisément quelles modifications devez vous apporter aux fichier layout.xml et au fichier java ?

Références

Généralités sur les interfaces graphiques

- Actions utilisateur :
  Controls http ://developer.android.com/guide/topics/ui/controls.html
  Events http ://developer.android.com/guide/topics/ui/ui-events.html
- Ressources :
  Accès
  http ://developer.android.com/guide/topics/resources/accessing-resources.html
  Types
  http ://developer.android.com/guide/topics/resources/available-resources.html

Références

Views et ViewGroups

- Layout :
  http ://developer.android.com/guide/topics/ui/declaring-layout.html
  http ://developer.android.com/guide/topics/ui/layout/linear.html
  http ://developer.android.com/reference/android/widget/LinearLayout.html
  http ://developer.android.com/reference/android/widget/GridLayout.html
- Button :
  http ://developer.android.com/guide/topics/ui/controls/button.html
  http ://developer.android.com/reference/android/widget/Button.html
- EditText :
  http ://developer.android.com/reference/android/widget/EditText.html
- TextView :
  http ://developer.android.com/guide/topics/ui/controls/text.html
  http ://developer.android.com/reference/android/widget/TextView.html

Prochain TP : cycle de vie d'une activité