Android - TP1 : les premiers pas

- Récupérez sur Chamilo le projet sur lequel vous allez travailler : AndroidTP.zip, placez le dans un dossier R4_Real_11.
- Attention n'utilisez pas le dossier Android à la racine de votre homedir. Il est utilisé par Android Studio et pointe sur un dossier tmp.
- Lancez l'IDE Android Studio et ouvrez le projet
  - Sur le terminal, tapez « **android-studio &** ».
  - Open project (le projet AndroidTP téléchargé)
  - Question au chargement Android SDK Manager : **VOIR prise en main android studio**
- Vous trouverez sur le web toute la documentation nécessaire pour ce module
  - Documentation Java : http://docs.oracle.com/javase/8/docs/api
  - Guide User Interface Android : http://developer.android.com/guide/topics/ui/index.html
  - Android référence : http://developer.android.com/reference/android/package-summary.html

Device ou émulateur Android ?

Nous vous conseillons pour les stations linux des salles informatiques d'utiliser un device Android (smartphone ou tablette) pour réaliser vos TPs. L'utilisation d'un device sera moins gourmande en mémoire que l'émulateur. De plus, vos tests seront plus proches de la réalité d'usage. N'oubliez pas que l'on testera vos projets sur device !

Si vous utilisez VOTRE DEVICE : vous devez activer l'option "Débuggage USB" qui se trouve dans le menu caché "Options pour les développeurs". Pour activer ce menu caché : Rendez-vous dans les options du téléphone > Cliquez sur "A propos du téléphone" > Cliquez 7 fois sur "Numéro de build"

Tablettes du DEPARTEMENT INFORMATIQUE : le département vous prête des tablettes pour la durée de la séance.

EMULATEUR : si vous souhaitez continuer à développer chez vous mais que vous ne possédez pas de device Android, vous pouvez utiliser l'émulateur d'android studio. Reportez vous au manuel sur Chamilo pour son installation et son exécution (document prise-en-main-android-studio ~~R4.Real.11-Manuel.AndroidStudio~~)

Prise en main d'Android Studio

Dans le projet androidTP, pour chaque exercice, des fichiers incomplets vous sont fournis à vous de les compléter.

AIDE : Vous trouverez sur Chamilo un manuel regroupant les principaux usages d'Android Studio (document prise-en-main-android-studio ~~R4.Real.11-Manuel.AndroidStudio~~) et une FAQ (document R4.Real.11-FAQ)

Exercice 1 : Première activité

Objectifs : créer une interface utilisateur et un abonnement d'événement

Pour réaliser cet exercice, vous aurez à modifier `Exercice1Activity.java` et `activity_exercice1.xml` (respectivement la classe représentant une activité et sa vue associée).

Scénario : un utilisateur entre son prénom (par exemple Jérôme) dans un objet de type `EditText`. Lors du clic sur le bouton valider, le texte « Hello world ! » sera remplacé par « Hello Jérôme ! » (voir figure 1).

Figure 1

Etape 1 : **créer** l'interface graphique de la figure 1 dans le fichier `activity_exercice1.xml` en respectant l'arborescence suivante :

- `LinearLayout`
  - `TextView`
  - `EditText` (`id : exercice1_prenom`)
  - `Button`
  - `TextView` (`id : exercice1_hello`)

IMPORTANT: Le fichier `activity_exercice1.xml` possède déjà un gestionnaire d'agencement (Layout en android) : un `LinearLayout`. Pensez à ajouter un attribut d'orientation : `android:orientation="vertical"` pour orienter votre layout. Pour plus de détails, se référer au cours 1 et aux liens vers les guides, ci-après le guide pour le `LinearLayout` : https://developer.android.com/develop/ui/views/layout/linear

ATTENTION lorsque que vous crée une nouvelle activité (Empty Activity), le layout de base est de type `android.support.constraint.ConstraintLayout` (surtout pour être utilisé par l'éditeur graphique). Dans un premier temps, nous souhaitons que vous utilisiez `LinearLayout` et `RelativeLayout` pour vous familiariser avec la construction d'un layout sans aide extérieur.

Vous modifierez le fichier `activity_exercice1.xml` directement sur le fichier XML. Par exemple, ci-dessous un exemple d'objet graphique de type `TextView` :

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/exercice1_text"
    android:gravity="center"/>
```

La valeur `match_parent` permet de prendre toute la place laisser par le parent, `wrap_content` prend juste la place nécessaire. La valeur `@string/exercice1_text` représente une variable présente dans le fichier `res/values/strings.xml` associée à une chaîne de caractères (si la variable n'est pas présente dans le fichier et ça sera le cas pour vous, l'IDE vous proposera de créer la variable et sa chaîne associée, cliquez sur l'ampoule

rouge et laissez-vous guider). `android:gravity`[1] est une propriété de l'objet de Type `TextView` (représente la méthode `setGravity()`) qui permet de spécifier l'alignement du texte dans la vue.

Nous vous conseillons de tester les différentes propriétés et valeurs des objets graphiques et de vérifier les résultats sur l'affichage « split » proposée par Android Studio. Référez-vous aussi à la documentation pour connaître les propriétés des classes[2].

Etape 2 : **réaliser** la partie événementielle de cette activité dans `Exercice1Activity.java`.

Pour rappel : un utilisateur entre son prénom (par exemple Jérôme) dans un objet de type `EditText`. Lors du clic sur le bouton valider, le texte « Hello world ! » sera remplacé par « Hello Jérôme ! » (voir figure 1).

Première chose à faire, **créer une écoute (abonnement)** sur l'objet qui va provoquer un événement click, ici le bouton Valider. Par exemple ci-après, on ajoute au bouton d'identifiant (id) `exercice1_valider` une méthode `exerciceValider` qui « écoutera » les événements que ce bouton produira en `onClick`.

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/exercice1_valider"
    android:onClick="exercice1Valider"/>
```

Deuxième chose à réaliser, **avoir accès aux composants graphiques** de l'interface qui seront à modifier dans le fichier Java :

- dans le XML, donner un id à votre objet de type `TextView` (dernier objet du `LinearLayout`) :

  ```
  android:id="@+id/exercice1_hello"
  ```

- dans le Java, récupérer cet objet en utilisant l'id sous la forme `R.id.exercice1_hello` et la méthode `findViewById(…)` :

  ```
  TextView helloView = findViewById(R.id.exercice1_hello);
  ```

Troisième chose, compléter la méthode d'écoute pour **modifier cet objet graphique** :

```
helloView.setText("hello " + prenomView.getText() + " !");
```

Aide : L'objet `prenomView` sera l'objet de type `EditText` présent dans le `LinearLayout` (`android:id="@+id/exercice1_prenom"`) que l'utilisateur doit remplir. Récupérer cet objet de la même façon que l'objet `helloView`.

Vous pouvez vérifier que l'utilisateur a donné un prénom en utilisant la méthode `TextUtils.isEmpty(…)` :
https://developer.android.com/reference/android/text/TextUtils#isEmpty(java.lang.CharSequence)

Tester votre application en lançant la commande de l'IDE Run > Run 'app'. Cette commande chargera votre application (.apk) sur votre device.

---

[1] http://developer.android.com/reference/android/widget/TextView.html#setGravity(int)

[2] http://developer.android.com/reference/android/package-summary.html
http://developer.android.com/guide/topics/ui/index.html

Pour résumer la méthode d'écoute `exercice1Valider` ressemblera à :

```
public void exercice1Valider(View view) {

    // On récupére les objets de l'arbre graphique (à l'aide de leur id)
    TextView helloView = findViewById(R.id.exercice1_hello);
    EditText prenomView = findViewById(R.id.exercice1_prenom);

    // On affiche le message si un prénom a été défini
    if (!TextUtils.isEmpty(prenomView.getText())) {
        helloView.setText("Hello " + prenomView.getText() + " !");
    }
}
```

ATTENTION, la méthode `onClick` est **deprecated**, c'est-à-dire que cette méthode doit être évitée dans un nouveau code (et enlevée dans les anciens). Modifier votre code pour utiliser un listener spécifique (classe anonyme, transparent 29-30 du cours 1) à la place du `onClick(…)` dans votre XML.

Premières réalisations

Exercice 2 : Question-réponse

Objectifs : créer une interface utilisateur, rechercher des objets graphiques adaptés aux besoins, et créer un abonnement d'événement.

Pour réaliser cet exercice, vous aurez à modifier `Exercice2Activity.java` et `activity_exercice2.xml` (respectivement la classe représentant une activité et sa vue associée).

Scénario : Cette activité proposera une question et 3 choix possibles (voir figure 2). Une fois la réponse choisie, l'utilisateur validera pour vérifier s'il a la bonne réponse. La bonne réponse sera toujours la première.

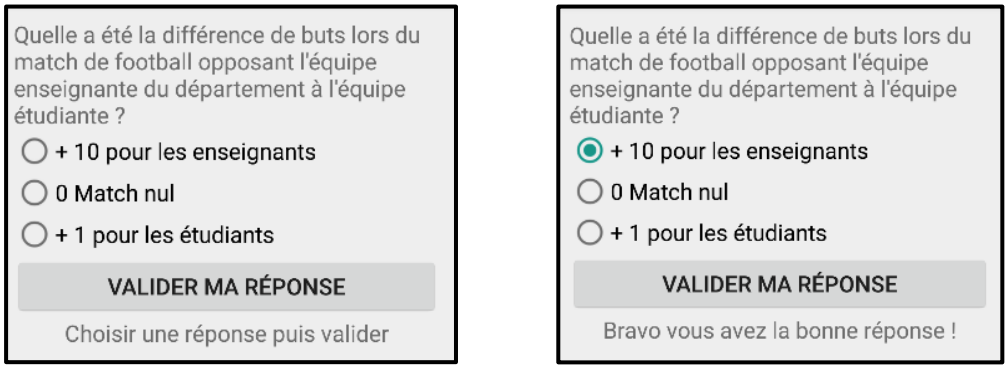

Figure 2

Étape 1 : créer l'interface graphique de la figure 2 dans le fichier `activity_exercice2.xml`. L'outil de design proposé par Android Studio vous permettra de connaître toute la palette d'objets graphiques à votre disposition.

CONSEIL SUR L'OUTIL DESIGN : Vous pouvez utiliser l'outil de design **MAIS** sachez qu'il générera le fichier XML en ajoutant peut être des choses que vous ne comprenez pas ! Cet outil est très puissant et doit donc être utilisé avec prudence. Nous vous conseillons de créer les fichiers XML à la main dans un premier temps puis quand vous aurez bien compris le fonctionnement des objets graphiques (layout, etc.) d'utiliser l'outil de design.

AIDE : placez les objets de Type `RadioButton` dans un container de Type `RadioGroup`.

Étape 2 : réaliser la partie événementielle de cette activité dans le fichier `Exercice2Activity.java`.

AIDE : vous pouvez comparer des composants graphiques entre eux. Par exemple, pour vérifier que l'utilisateur à la bonne réponse vous pouvez comparer les identifiants de l'objet de type `RadioButton` sélectionné (en demandant à l'objet de type `RadioGroup`) et l'objet de type `RadioButton` pourtant la bonne réponse :
`if (radioGroupView.getCheckedRadioButtonId() == R.id.exercice2_bonne_reponse) {…`

Autre solution possible, utilisez la méthode `isChecked()` d'un objet de type `RadioButton` :
`if (bonneReponseView.isChecked()) {`

Étape Bonus (pour les plus aventureux/avancés) : modifier pour que les réponses soient mélangées à chaque lancement de l'activité. Une solution possible :

1. Créer les données dans l'activité :
```
// DATA
String bonneReponse = "+10 pour les enseignants";
String mauvaiseReponse1 = "0 match nul";
String mauvaiseReponse2 = "+1 pour les étudiants";
```
2. Créer une liste contenant les données ci-dessus et la mélanger (utiliser la méthode `Collections.shuffle(reponses)`)
3. Remplir les `RadioButton` avec les données (utiliser la méthode `radioGroupView.getChildAt(i)` pour les récupérer puis les modifier avec la méthode `setText(…)`)
4. Comparer la réponse sélectionnée par l'utilisateur avec la bonne réponse :

```
if (radioButtonSelectionne.getText().equals(bonneReponse)) {
```

Exercice 3 : Pierre-Feuille-Ciseaux

***Objectifs* :** créer une interface utilisateur complexe, créer des abonnements et utiliser un modèle de données.

Pour réaliser cet exercice, vous aurez à modifier `Exercice3Activity.java` et `activity_exercice3.xml` (respectivement la classe représentant une activité et sa vue associée).

En vous inspirant de l'exercice 2, réaliser le jeu pierre-feuille-ciseaux (voir figure 3). Pour plus d'information : http://fr.wikipedia.org/wiki/Pierre-feuille-ciseaux

IMPORTANT : Pour vous aider, les classes `Jeu` et `Resultat` vous sont données dans le projet dans le package `exercice3Data` (une archive TestJeu.zip vous est également fournie pour que vous puissiez tester ces classes sur le terminal – lire `TestJeu.java`). **A QUOI SERVENT CES CLASSES ?** La classe `Jeu` simule un pierre-feuille-ciseaux c'est le modèle du jeu (une partie si vous préférez). A vous d'utiliser un objet de type `Jeu` dans l'activité `Exercice3Activity.java`. La classe `Resultat` représente le résultat des jeux (nb de victoires, nb de défaites et nb d'égalités). A vous d'utiliser un objet de type `Resultat` dans l'activité `Exercice3Activity.java` pour enregistrer le résultat des différentes parties. En gros, l'activité est le contrôleur, les classes `Jeu` et `Resultat` le modèle et le XML la vue : **Modèle-Vue-Controleur MVC**. *Vous pouvez modifier ces classes si besoin.*

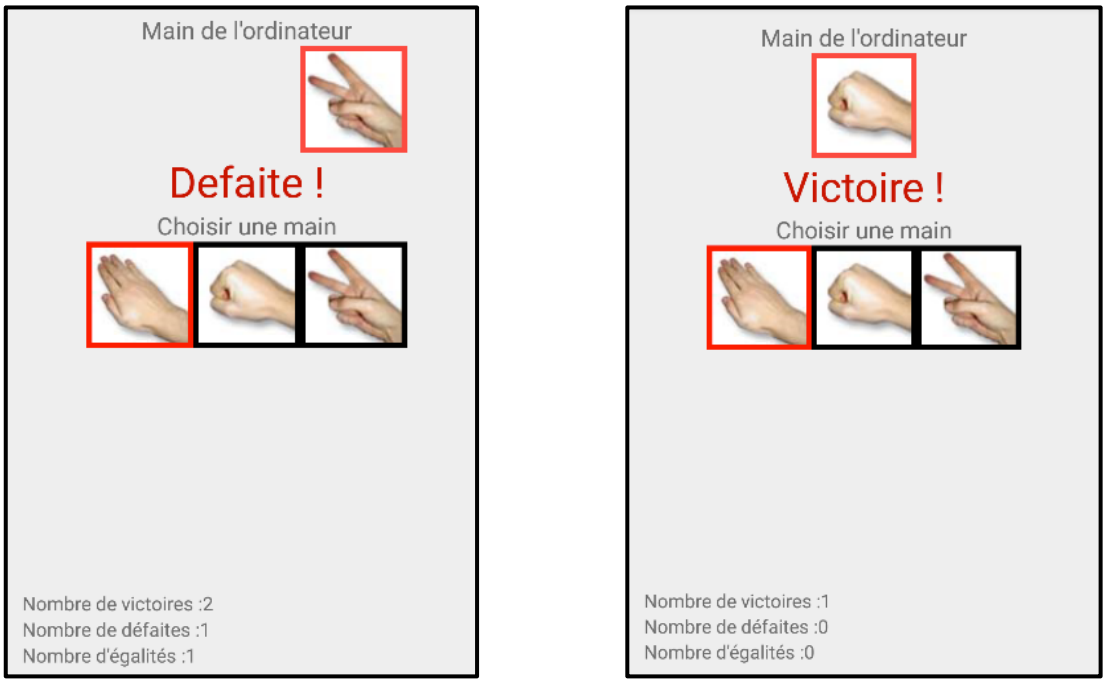

Figure 3

Etape 1 : **créer** l'interface graphique de la figure 3. Les images à utiliser sont sur Chamilo. Placez-les dans le dossier `res/drawable`.

- Les images sont clickable ;
- Les objets graphiques peuvent être affichés ou cachés avec l'attribut `visibility` ;
- Les bordures sont faites avec un background et des marges ;
- Un objet graphique de type `TextView` peut modifier son texte avec un texte ressources contenu dans le fichier strings.xml, par exemple « Victoire ! » ou « Défaite ! » ;

```
resultatView.setText(R.string.exercice3_victoire);
```

- L'affichage du nombre de victoires/défaites/égalités peut se faire en décomposant le texte et le nombre, donc deux `TextView` pour une ligne. Attention, si vous mettez un nombre en paramètre d'une méthode `setText(…)`, Android cherchera une ressources de type `R.id.quelquechose`. Pensez donc à transformer votre nombre en une chaîne de caractères avant !

AIDE : Pensez à utiliser des Layouts[3] pour agencer vos objets graphiques : `LinearLayout`, `RelativeLayout`, etc. Utiliser les propriétés `orientation`, `gravity`, etc.

Dans cet exercice, nous vous conseillons d'utiliser `RelativeLayout` comme layout principal et des `LinearLayout` pour les alignements d'images et des nombres de victoires, défaites et égalités. Voir un exemple de `RelativeLayout` dans le guide android :

https://developer.android.com/develop/ui/views/layout/relative

Etape 2 : **réaliser** la partie événementielle de cette activité.

IMPORTANT : inspirez-vous de l'exemple de code propre proposé dans le cours (derniers transparents).

AIDE : Un objet de Type `ImageView` est clickable. Donc on peut associer une méthode d'écoute à un objet de Type `ImageView`.

[3] Guide Layout : http://developer.android.com/guide/topics/ui/declaring-layout.html

Etape 3 : **réaliser** une vue spécifique pour le mode landscape/paysage (voir figure 4).

Créer une nouvelle vue XML pour cette activité :

- New > Android Resource File ;
- Ressource type, choisir layout ;
- Available qualifiers, choisir orientation ;
- Screen orientation, landscape ;
- Donner le même nom de fichier `activity_exercice3.xml` que la version « normale ».

Ce fichier sera automatique chargé par l'activité quand le device sera en landscape. Modifier donc ce fichier pour correspondre à la figure 4.

Figure 4

IMPORTANT : notez que les résultats sont effacés lors du changement de position du device. En effet, l'activity est recrée lors du changement. Nous verrons dans les prochaines séances comment enregistrer des informations et comment détecter les modifications survenues sur le device.