

# Android – TP2 : les premières intentions

- Continuez à travailler sur le projet AndroidTP récupéré au TP1.
- Attention n'utilisez pas le dossier Android à la racine de votre homedir. Il est utilisé par Android Studio et pointe sur le dossier tmp.
- Lancez l'IDE Android Studio et ouvrez le projet
  - IMPORTANT lire le fichier *IMPORTANT-LancementAndroidStudio3.5.pdf*
  - Sur le terminal, tapez « **android-studio-2022.1 &** ».
  - Open project (le projet AndroidTP téléchargé)
- 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>

**RAPPEL** : 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 !

**AIDE - LANCER VOTRE PROJET** : Si votre projet ne compile pas, faites un update de vos SDK tools et synchroniser votre projet avec gradle (voir la FAQ document R4.Real.11-FAQ)

## Prise en main des intentions

### Exercice 4 : Premières intentions

**Objectifs** : créer une nouvelle activité et une demande d'intention de lancement d'activité

Pour réaliser cet exercice, vous aurez à modifier `Exercice4Activity.java` et `activity_exercice4.xml` (respectivement la classe représentant une activité et sa vue associée) et à créer une nouvelle activité `HelloActivity.java` avec sa vue associée.



Figure 2 : HelloActivity

**Scénario :** un utilisateur écrit son prénom (par exemple Jérôme) dans un objet de type EditText (voir figure 1). Lors du clic sur le bouton valider (clic 1), un texte « Hello Jérôme! » apparaîtra sur une nouvelle activité (voir figure 2). Cette nouvelle activité proposera de changer de nom (clic 2 - retour à l'activité précédente) ou de choisir un autre exercice (clic 3 - retour à l'activité principale MainActivity).

**Etape 1 :** créer l'interface graphique de la figure 1 dans le fichier `activity_exercice4.xml`.

**Etape 2 :** créer une nouvelle activité `HelloActivity` : *File > New... > Activity > Empty Activity*. Dans *Activity name*, mettre `HelloActivity`. La vue XML associée (*Layout name*) aura comme nom `activity_hello`. Cliquer sur le bouton Finish. Créer l'interface graphique de la figure 2 dans le fichier `activity_hello.xml` en remplaçant `ConstraintLayout` par un `RelativeLayout`.

**Etape 3 :** réaliser la demande d'intention dans la classe `Exercice4Activity` permettant le lancement de la nouvelle activité `HelloActivity`. Pour ce faire :

1. Réaliser la partie événementielle du bouton Valider : un abonnement de l'événement click à ce bouton.
2. Dans la méthode `onClick` de l'abonnement, récupérer la chaîne de caractères donnant le prénom dans le composant graphique de type `EditText` :  

```
String prenom = prenomView.getText().toString();
```
3. Créer une intention pour la nouvelle activité `HelloActivity` :  

```
Intent intent = new Intent(Exercice4Activity.this, HelloActivity.class);
```
4. Ajouter la chaîne à l'intention pour transférer le prénom à la nouvelle activité :  

```
intent.putExtra(HelloActivity.PRENOM_KEY, prenom);
```

  
`PRENOM_KEY` est une constante de l'activité représentant le nom du paramètre. Ajouter l'instruction suivante comme attribut de classe dans `HelloActivity` :  

```
public static final String PRENOM_KEY = "prenom_key";
```
5. Lancement de la demande d'intention en utilisant la méthode `startActivity`  

```
startActivity(intent);
```

**Vérifier** à chaque étape que votre programme fait bien ce que vous voulez. Par exemple, dans cette étape valider le changement de fenêtre (la récupération du prénom se fera dans l'étape suivante).

**Rappel :** pour tester votre application en lançant la commande de l'IDE *Run > Run 'app'*. Cette commande chargera votre application (.apk) sur votre émulateur.

**Etape 4 :** récupérer le prénom passé par l'intention dans l'activité `HelloActivity`. Pour ce faire, ajouter l'instruction suivante dans la méthode `onCreate`. Pour rappel, cette méthode crée l'activité et affiche la vue associée.

```
String prenom = getIntent().getStringExtra(PRENOM_KEY);
```

**Remarque :** noter l'usage de la constante `PRENOM_KEY` de la classe `HelloActivity` représentant la clé (le nom) du paramètre envoyé.

Ensuite, ajouter ce prénom dans un composant de type `TextView` pour afficher « Hello prénom !!! ». Attention, la méthode `setId(...)` demande soit une chaîne de caractères soit un entier représentant l'identifiant d'une ressource dans le fichier `Strings.xml` dans le dossier `res/values`

**Petit plus :** vous pouvez ajouter un paramètre à une ressource du fichier `Strings.xml` par exemple :

```
<string name="hello_text">Hello %1$s !</string> // %1$s étant un paramètre de type String
```

Avec comme utilisation :



```
helloView.setText(getResources().getString(R.string.hello_text, prenom));
```



[https://developer.android.com/reference/android/content/Context#getString\(int,%20java.lang.Object...\)](https://developer.android.com/reference/android/content/Context#getString(int,%20java.lang.Object...))

**Etape 5 : réaliser** le retour à l'activité Exercice4Activity. L'activité HelloActivity propose à l'utilisateur un bouton « Changer de prénom ». Lorsque ce bouton est cliqué, l'application revient à l'activité Exercice4Activity. Pour réaliser ce retour, l'activité HelloActivity est arrêtée et enlevé de la pile d'activité ce que provoque la réapparition de l'activité Exercice4Activity. Pour ce faire, ajouter à la méthode onClick lors de l'abonnement au bouton l'instruction suivante : `finish()` ;



**Etape 6 : réaliser** le retour à l'activité MainActivity. L'activité HelloActivity propose à l'utilisateur un bouton « Choisir un autre exercice ». Lorsque ce bouton est cliqué, l'application revient à l'activité MainActivity. Pour réaliser ce retour, l'activité HelloActivity fait une demande d'intention pour l'activité MainActivity :



```
Intent intent = new Intent(HelloActivity.this, MainActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
```

Vous remarquerez que le flag `Intent.FLAG_ACTIVITY_CLEAR_TOP` est utilisé pour éviter l'empilement des activités dans la pile. Enlever ce flag et tester votre application pour comprendre son intérêt : Cliquer plusieurs fois sur « Choisir un autre exercice » et ensuite cliquer sur le bouton retour  ou  pour vérifier l'empilement des activités

**Etape 7 : Informer** sur le retour HelloActivity vers Exercice4Activity : Changer de prénom ou bouton retour  ou . Prendre exemple sur les transparents 13 à 17 du cours 2.

#### Scénario :

- Quand l'utilisateur clique sur le bouton « changer de prénom » dans l'activité HelloActivity, lors du retour sur l'activité Exercice4Activity un message apparaît « nouveau prénom » et faire un clear de l'editText.
- Si l'utilisateur clique sur le bouton retour  ou  dans l'activité HelloActivity, lors du retour sur l'activité Exercice4Activity un message apparaît « bouton back » et l'editText conserve le prénom précédent.

1. Modifier le lancement de la demande d'intention de la classe Exercice4Activity en utilisant un objet de type `ActivityResultLauncher`. Cet objet permettra d'écouter l'arrêt de l'activité HelloActivity (voir transparents 14, 15 et 17 du cours 2). Remplacer l'instruction `startActivity(intent);` de l'activité Exercice4Activity par l'instruction : `activityResultLauncher.launch(intent);`
2. Le `resultCode` permettra de déterminer le choix utilisateur lors de l'arrêt de l'activité HelloActivity soit bouton retour  ou  soit le bouton « changer prénom ». Pour ce dernier cas, ajouter à la classe HelloActivity avant l'instruction arrêtant l'activité `finish()` ; l'instruction : `setResult(RESULT_OK);` (voir transparent 16 du cours 2).

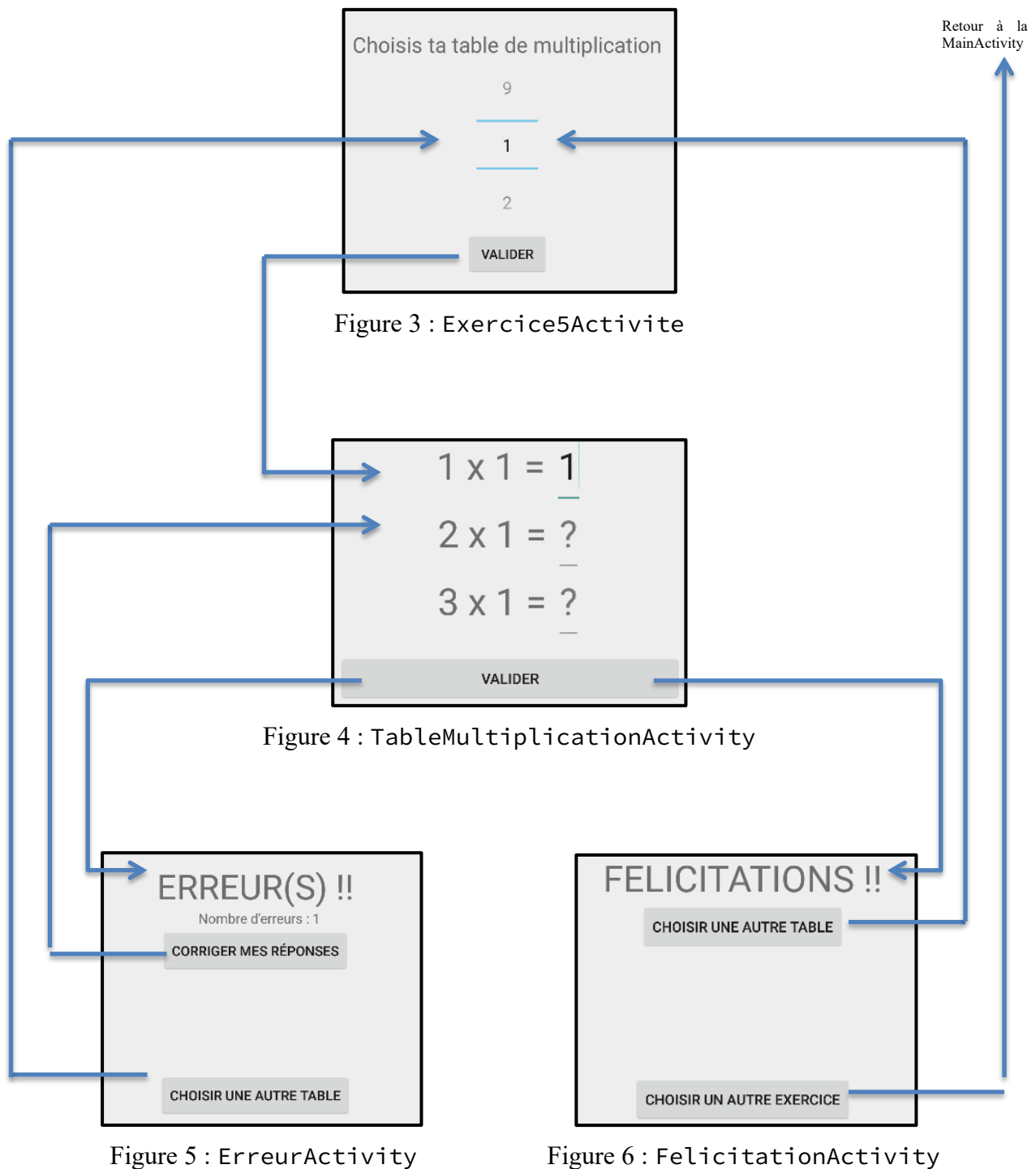
## Réalisation

### Exercice 5 : S'entraîner aux tables de multiplication

**Objectifs :** créer des activités, gérer des événements, créer des intentions, générer une interface utilisateur

Pour réaliser cet exercice, vous aurez à modifier `Exercice5Activity.java` et `activity_exercice5.xml` (respectivement la classe représentant une activité et sa vue associée) et créer de nouvelles activités pour répondre au besoin.

**Scénario :** l'utilisateur choisit la table de multiplication (**NumberPicker**) avec laquelle il souhaite s'entraîner (voir figure 3). Après validation, il est envoyé sur une activité proposant la table et les parties à remplir (voir figure 4). Après validation, l'utilisateur est soit renvoyé sur une activité donnant le nombre d'erreurs et proposant de corriger les réponses ou choisir une autre table (voir Figure 5), soit, si aucune erreur, renvoyé sur une activité le félicitant et proposant de choisir une autre table ou de choisir un autre exercice (voir Figure 6).



## Étapes Bonus pour les plus aventureux/avancés

### Afficher les erreurs de l'utilisateur en rouge lors du retour sur correction

**Scénario** : lors du clique sur le bouton « corriger », l'utilisateur verra en rouge ses erreurs pour l'aider dans sa correction (voir figure 7 ci-dessous). Si l'utilisateur revient sur l'activité TableMultiplicationActivity avec un simple retour  $\triangleleft$  ou  $\leftarrow$ , il ne verra pas ses erreurs en rouge.

**Aide** : utiliser un objet de type ActivityResultLauncher. Cet objet permettra d'écouter l'arrêt de l'activité ErreurActivity (voir transparents 14, 15 et 17 du cours 2)

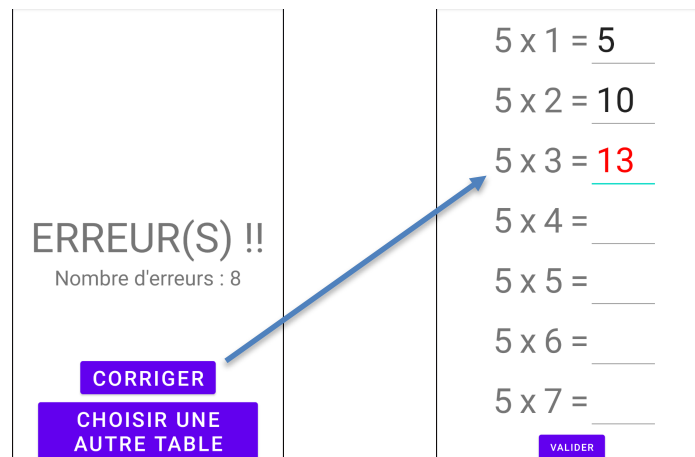


Figure 7 : afficher les erreurs de l'utilisateurs en rouge lors du retour sur correction

### 2 modes : entraînement ou évaluation

**Scénario** : proposer à l'utilisateur 2 modes pour réaliser sa table de multiplications Soit le mode entraînement qui affichera la table de multiplication choisie dans l'ordre (c'est ce que vous avez déjà réalisé), soit le mode évaluation qui affichera la table de multiplication choisie mélangée (voir figure 8).

**Aide** : si vous ne l'avez pas encore fait, réaliser un modèle représentant une table de multiplication (comme les classes Jeu et Résultat pour le jeu Pierre-Feuille-Ciseaux). Nous vous conseillons de créer deux classes TableDeMultiplication et Multiplication pour ce faire. La classe TableDeMultiplication se chargera de mélanger les multiplications si l'utilisateur est en mode évaluation. Elle permettra aussi de recevoir les réponses de l'utilisateur et donner ainsi le nombre d'erreurs. Attention, les classes modèles ne doivent contenir aucune référence aux briques graphiques de l'interface.

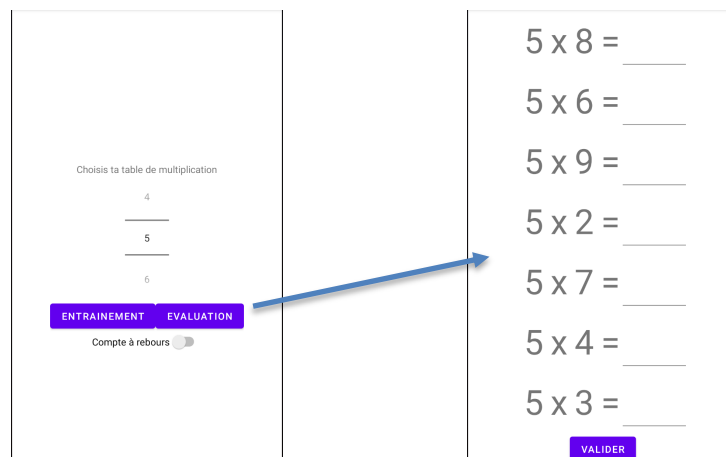


Figure 8 : mode évaluation

### Proposer un compte à rebours à l'utilisateur

**Scénario :** l'utilisateur pourra choisir d'activer l'exercice avec un compte à rebours de 30 secondes ou non. Si l'utilisateur choisit de l'activer, un compte à rebours apparaîtra en haut à droite de `TableMultiplicationActivity` (voir figure 9). A la fin du compte à rebours, validation automatique des réponses et donc l'utilisateur sera redirigé vers `ErreurActivity`.

**Aide :** pour réaliser un compte à rebours, utilisez la classe `CountDownTimer` proposée par le SDK d'android. Lors du lancement d'un objet de type `CountDownTimer`, un compte à rebours sera lancé sur un processus à part de votre application. L'objet de type `CountDownTimer` sera prévenu à un intervalle de temps régulier (déterminé par vous) du temps restant à l'aide de la méthode `onTick(...)` et lors de la fin du compte à l'aide de la méthode `onFinish()` (de la même manière qu'un abonnement à une brique graphique source d'événement).

À voir <https://developer.android.com/reference/android/os/CountDownTimer>

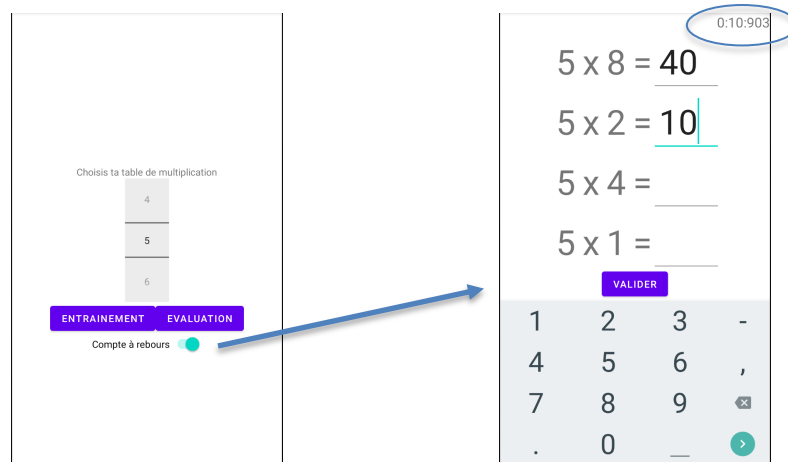


Figure 9 : compte à rebours