**Faculté des sciences Ain Chock 2021-2022**

# Master BIG DATA & Cloud Computing

Développement sous Android

TP2

**Cycle de vie d’une activity**

*NB : concevoir un filtre dans la vue logcat à partir d'un même champs « tag » pour ces opérations de log. Cela sera plus simple pour différencier les logs systèmes de celle de votre application.*

* Ajouter la méthode « *onDestroy* » à l'activité principal
* Instancier 'log.i' dans les méthodes « *onCreate* » & « *onDestroy* »
* Visualiser (vue 'logcat') les messages précédants lors du lancement et de l'arrêt de l'application
* Répéter ces opérations avec les méthodes « *onPause* » & « *onResume* »
* Relancez l'application, émuler un appel (vue « emulator control »), retrouver le focus de votre application (en raccrochant !). Observer le logcat

=> Bien faire le parallèle entre le logcat avec le cycle de vie d'une activité (perte de focus / retour de focus)

* Tuer l'application (le processus donc) dans « *onDestroy* » (librairie java standard)

*PS : Tourner l'écran en « landscape » (dans l'emulateur ctrl+F12). Que se passe-t-il ? Pourquoi ?* – trouver le moyen pour la tuer 'proprement' … lorsque l' « activity » sait qu'elle « *est finie* » ;)

# Création de Widget / UI et Interaction avec le code

Dans cette partie nous allons faire abstraction de la programmation de widget en java en travaillant directement sur la template xml (**content\_main**.xml ici).

* Modifier la template (vous resterez toujours avec le RelativeLayout comme parent) afin de rajouter 2 champs de saisies et un bouton afin d'obtenir ceci :

*conseils :*

* *Travailler avec le layout et non l'XML brut.*
* *Changer les identifiants (id) des champs de saisies et du boutton afin de les retrouver facilement dans le code.*
* *Changer d'autres propriétés des différents widgets afin d'obtenir le résultat*
* Créer le handler (dans la methode « onResume ») du bouton ENVOYER. Afficher cet événement dans le logcat (recrée un nouveau tag + filtre)
* Créer une fonction « envoie\_message » dans l'activité permettant de récupérer les champs de saisies et les afficher dans le logcat lors de l'appui du bouton ENVOYER
* Créer un Toast à la fin de la fonction en écrivant que 'le message est prêt à être envoyé’
* Créer un handler de menu pour l'activité (méthode « onCreateOptionsMenu ») … en « *ajoutant* » un « *menu* » intitulé ' envoyer ' ;)
* Créer le sélectionneur d'item de menu (méthode « onOptionsItemSelected ») qui appellera la fonction « envoie\_message » crée précédemment ( uniquement sur sélection du menu 'envoyer' )

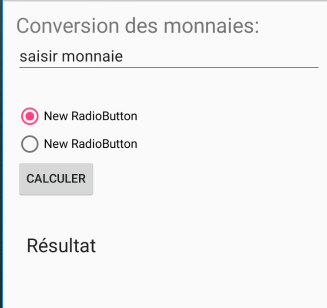
# Communication avec d'autres applications

– Créer un intent de type ACTION\_SEND dans la fonction « envoie\_message ». Préparer dans celui-ci les différents champs saisies. Démarrer le en lui demandant de « creerunchoix ». ;)

Composants Android

Ce TP a pour objectif de vous initier aux différents composants importants d’Android. Nous allons dans ce TP réaliser progressivement une petite application de conversion.

Créer un nouveau projet intitulé *Conversion*. Créer les éléments nécessaires pour que l’interface soit comme la figure suivante :



**1**. Associer un comportement à un bouton

1 Créer une méthode dans le code Java de l’activité qui définit le comportement du bouton. On l’appellera

public void auClicMethode(View v){…}

Cette méthode doit obligatoirement être publique, retourner *void* et avoir un paramètre de type *android.view.View*.

2. Créer un bouton dans le fichier *layout* en utilisant la partie graphique.

3. Modifier son identifiant et son texte selon votre choix

4. Dans le code xml de votre bouton, ajouter l’attribut :

android:onClick = "@string/auClic"

5. Créer dans le fichier *strings.xml* un nouveau string dont le nom est *auClic* et la valeur est *auClicMethode* (qui est le nom de la méthode que vous avez créé dans 1. )

Créer une méthode appelée *convertir* et l’associer au bouton *Convertir* de votre interface.

**2. Gérer les boutons radios**

Un bouton radio est un bouton à deux états qui peut être soit coché (*checked*) ou décoché *(unchecked)*. Les boutons radios sont en général utilisés dans un groupe *RadioGroup*. Au sein d’un même groupe, un seul bouton radio peut être coché.

Pour gérer l’état d’un bouton radio, il faut suivre les étapes suivantes :

1. Créer un attribut de type *RadioButton* dans votre activité (par exemple *radio1*).

2. L’associer au bouton radio approprié de votre interface en utilisant la méthode *findViewById*.

3. Pour tester l’état de votre bouton radio, appeler la méthode *isChecked()*. Par exemple :

if (radio1.isChecked() ){

//traitement

}

1. Créer deux méthodes : dhToEuro et euroToDh, qui prennent de convertir une valeur en entrée :

private float dhToEuro(float valeurDh) {

return \*\*\*\*;

}

private float euroToDh(float valeurEuro) {

return \*\*\*\*;

}

2. Implémenter la méthode *convertir* pour qu’elle fasse la conversion nécessaire, selon le bouton radio qui est coché.

Mettre le résultat dans le champ de texte *Resultat*.

*Indication : La valeur lue dans le champs de saisie (ici appelé* ***edt****) doit être convertie en float pour être manipulée. Pour cela,*

*utiliser le code suivant :*

EditText edt = (EditText) findViewById(R.id.edit\_float);

float number = Float.valueOf(edt.getText().toString());

*D’autre part, pour extraire la chaîne de caractères associée à une variable float (appelée ici* ***floatVar****), utiliser le code suivant :*

String s = String.valueOf(floatVar) ;

**3. Gérer les cases à cocher**

Tout comme les boutons radio, les cases à cocher ont deux états : coché ou décoché. Cependant, on peut avoir plusieurs cases qui sont cochées en même temps, et elles sont la plupart du temps indépendantes.

Pour gérer l’état d’une case à cocher, il faut suivre les étapes suivantes :

1. Créer un attribut de type *CheckBox* dans votre activité (par exemple *check1*).

2. L’associer à la case à cocher appropriée de votre interface en utilisant la méthode *findViewById*.

3. Pour tester l’état de votre case à cocher, appeler la méthode *isChecked()*. Par exemple :

if (check1.isChecked() ){

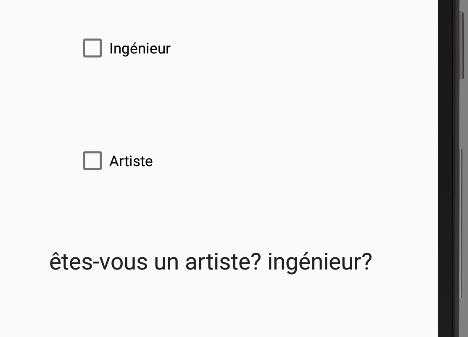
//traitement

}

4. Pour modifier l’état de la case à cocher, utiliser la méthode *setChecked(boolean etat)*. Par exemple :

check1.setChecked(false) ; //pour décocher la case

check1.setChecked(true) ; //pour cocher la case



**4. Menu contextuel**

Pour définir un menu contextuel, qui sera déclenché suite à un long clic sur un élément (que j’appelle ici *element :*

il peut être un bouton, un textView, ou même le layout en entier) :

1. Créer un attribut pour l’élément auquel on veut ajouter un menu contextuel, et l’associer à l’élément graphique avec la méthode *findViewById(…).*

2. Ajouter un écouteur pour le long clic, de la même manière que nous avons ajouté un écouteur pour le clic dans le TP

précédent. Cet écouteur( *Listener)*, va ordonner l’affichage du menu contextuel quand on fait un long clic sur *element* :

element.setOnLongClickListener(new OnLongClickListener() {

@Override

public boolean onLongClick(View v) {

v.showContextMenu();

return false;

}});

element.setOnCreateContextMenuListener (this) ;

3. Indiquer le comportement de ce menu. Pour cela, générer la méthode *onCreateContextMenu* dans votre activité. Elle est

appelée quand un menu contextuel va être affiché.

4. Dans le corps de cette méthode, on peut ajouter des menus. Pour cela, utiliser la méthode :

menu.add(groupID, itemID, ordre, "nom du menu") ;

Cette méthode permet de créer un nouveau menu. Les paramètres nécessaires sont les suivants :

--‐‑ *groupID* : identifiant du groupe. Il est possible de regrouper les éléments, mais dans notre cas, on n’en a pas

besoin, on lui donne donc la valeur 0 ;

--‐‑ *itemID* : identifiant de ce menu. Il nous sera utilise pour identifier ce menu parmi les autres. On doit donner

un identifiant différent à chaque menu (1, 2, 3… par exemple).

--‐‑ *ordre* : associer un ordre d’affichage au menu. On donnera toujours la valeur 0.

--‐‑ *Nom du menu* : chaîne qui représente le titre du menu.

5. Nous devons dire quel est le comportement à faire quand on clique sur cet élément du menu. Pour cela, générer la

méthode :

public boolean onContextItemSelected(MenuItem item){…}

Dans le corps de cette méthode, indiquer le comportement à adopter, si on clique sur le menu 1 ou 2 :

switch(item.getItemId()){

case 1:

//traitement 1

case 2:

//traitement 2

}

Créer un menu contextuel sur les deux boutons radios. Il doit contenir deux menus :

1. « *Taux dh --‐‑> euro* » : affiche dans un Toast le taux de conversion du dh vers l’euro.

2. « *Taux euro --‐‑> dh* » : affiche dans un Toast le taux de conversion de l’euro vers le dh.