

Création d'application Android

Partie 1

Objectif :

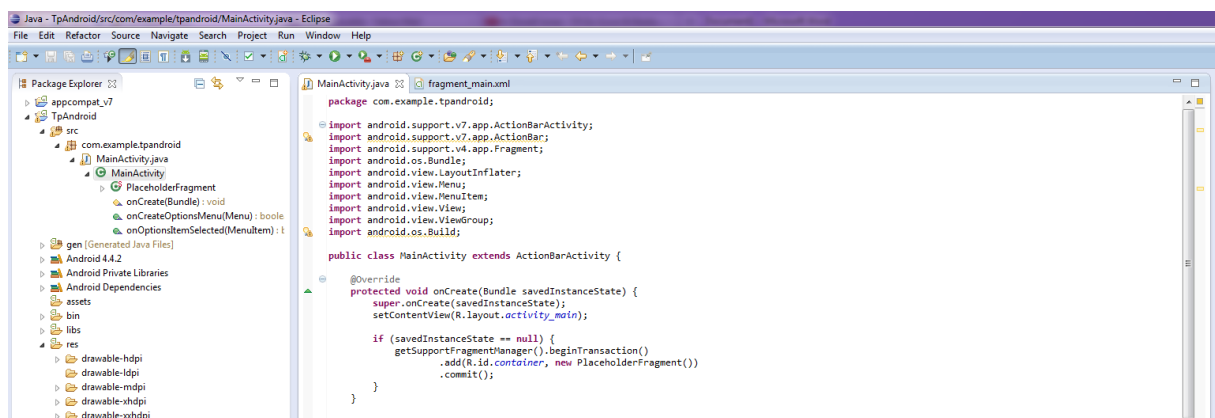
L'objectif de cette partie est d'apprendre les bases permettant de créer et lancer des applications Android.

Créer une nouvelle application avec eclipse :

Les applications Android sont développées en xml et Java. Android est un **système d'exploitation** à destination des terminaux mobiles.

File-> New-> Android ApplicationProject, nommez votre application et cliquez Next jusqu'à la création de votre projet.

Une fois créée vous obtenez un programme de base que vous trouvez dans MainActivity



La première ligne indique le package où se trouve votre application, puis vous avez les bibliothèques nécessaires au script : des "import" de package **android** (des package de base qui peuvent être utiles à votre application).

Vous avez une déclaration de la classe de base MainActivity qui dérive de la classe **Activity** ou **ActionBarActivity**. Une Application Android est un ensemble de fenêtre appelée "activité", les activités héritent de la classe **Activity**.

La méthode **onCreate(Bundle savedInstanceState)**, est la première méthode lancée au démarrage ou relancement d'une application, le paramètre de type **Bundle** indique l'état de l'application. A l'intérieur de cette méthode vous devez définir ce qui doit être créé à chaque démarrage **super.onCreate(savedInstanceState);** (super.onCreate fait appel à onCreate de la classe parente "ActionBarActivity" grâce au terme "super").

setContentView(R.layout.activity_main); applique " R.layout.activity_main" à la vue courante. Votre interface graphique est défini dans le fichier activity_main.xml que nous verrons dans la partie "création d'interface graphique".

Affichage d'un texte :

Dans cette partie vous allez rajouter quelques lignes de code pour afficher un texte. Sur Android, tous les éléments sont basés sur la classe **View**.

Le composant « View » est l'unité de base de l'interface graphique. Une interface graphique pour Android est constituée uniquement de vues qui représentent son contenu visuel avec qui nous pouvons interagir.

Les vues héritent de la classe **View**, vous les trouvez dans le package **android.view.View**.

Pour créer un champ texte il faut importer la bibliothèque correspondante
`import android.widget.TextView;`

Rajoutez un **TextView**

```
TextView message = new TextView (this) ;
```

Donnez le texte à afficher


```
message.setText("Mon application Android");
```

Appliquez le **TextView** à la vue courante


```
setContentView (message) ;
```

Lancement de l'application :

Afin de lancer votre application il faut que vous créez un téléphone virtuel qui vous permettra de visualiser le contenu de votre interface graphique.

Cliquer sur l'icone  "Android Virtual Device Manager ", le gestionnaire d'AVD s'ouvre. Cliquez sur "New", vous devez renseigner les différentes informations demandées : le nom représente le nom de l'AVD, **Target** : La version d'Android installée sur le téléphone virtuel. Les *targets* (cibles) disponibles ici sont celles installées grâce à l'exécutable "SDK Manager.exe" du SDK, **SD Card** : représente la taille de votre carte SD virtuelle ou bien un fichier, **Skin** : Pour un téléphone standard, choisissez "Default (HVGA)", **Hardware** : laissez le par défaut, cliquez sur OK pour finaliser la création de votre mobile virtuel.

Exécution de l'application :

Pour tester votre programme, lancez l'application dans l'AVD en choisissant "Run" dans le menu "Run " ou cliquez sur , vous devez choisir votre AVD sur le quel l'application s'exécutera.

Partie 2 : création d'un convertisseur de distance

L'objectif de cette partie est de créer une application qui permet de convertir une distance d'une unité à une autre (mètre en pied,...). Pour cela vous devez créer une interface graphique qui va contenir les différentes vues dans un fichier xml, puis implémenter son fonctionnement (les différents événements sur les vues dans un programme java).

Création d'interface graphique :



L'interface graphique se crée à travers un fichier xml qui va contenir tous les widgets de votre interface . Les différents évènements sur les éléments vont être implémentés en Java.

Lors de la création de votre projet vous obtenez un fichier xml générer automatiquement.

Ouvrez le fichier `activity_main.xml` qui se trouve dans le répertoire `res/layout` (dans la partie gauche de l'interface eclipse).

Les deux premières lignes indiquent que le fichier xml est spécifique à Android

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

`<RelativeLayout>` ce nœud représente la racine du fichier xml, il englobe un nœud de type `TextView` qui représente une vue. C'est à l'intérieur de cette racine que vous pouvez rajouter les différents éléments qui constituent votre interface graphique tels que les widgets: `TextView`, `EditText`, `Button`, `CheckBox`, `RadioButton` et `RadioGroup`... , chaque widget possède un nombre d'attributs XML et de méthodes Java.

Pour votre convertisseur de distance créez 3 zones de texte pour indiquer le nom de l'application "convertisseur de distance", l'unité source et l'unité de destination.

2 `RadioGroup`, afin de pouvoir choisir l'unité actuelle et l'unité de conversion. Mettez 4 `RadioButton` dans chaque groupe pour les unités : mètre, kilomètre, pied, pouce.

Un éditeur de texte **EditText** un pour saisir la distance et un pour afficher le résultat.

Un bouton **Button** pour lancer la conversion.

Les évènements :

Rajouter le package `android.view.MotionEvent` pour gérer les évènements. Rajouter les différents évènements sur les vues dans votre programme java.

La méthode `int getCheckedRadioButtonId()` permet de récupérer l'identifiant du `RadioButton` qui est sélectionné dans un `RadioGroup`.

Editable `getText()` permet de récupérer le texte d'un `EditText`.

Quelques règles de base :

1 mètre = 3,2809 pieds, 1 pied = 0,3048 mètre

1 km = 1000 m , 1 m = 0.001 km

1 km = 39370.078740157 in, 1 in = 2.54E-5 km

1 m = 39.370078740157 in, 1 in = 0.0254 m