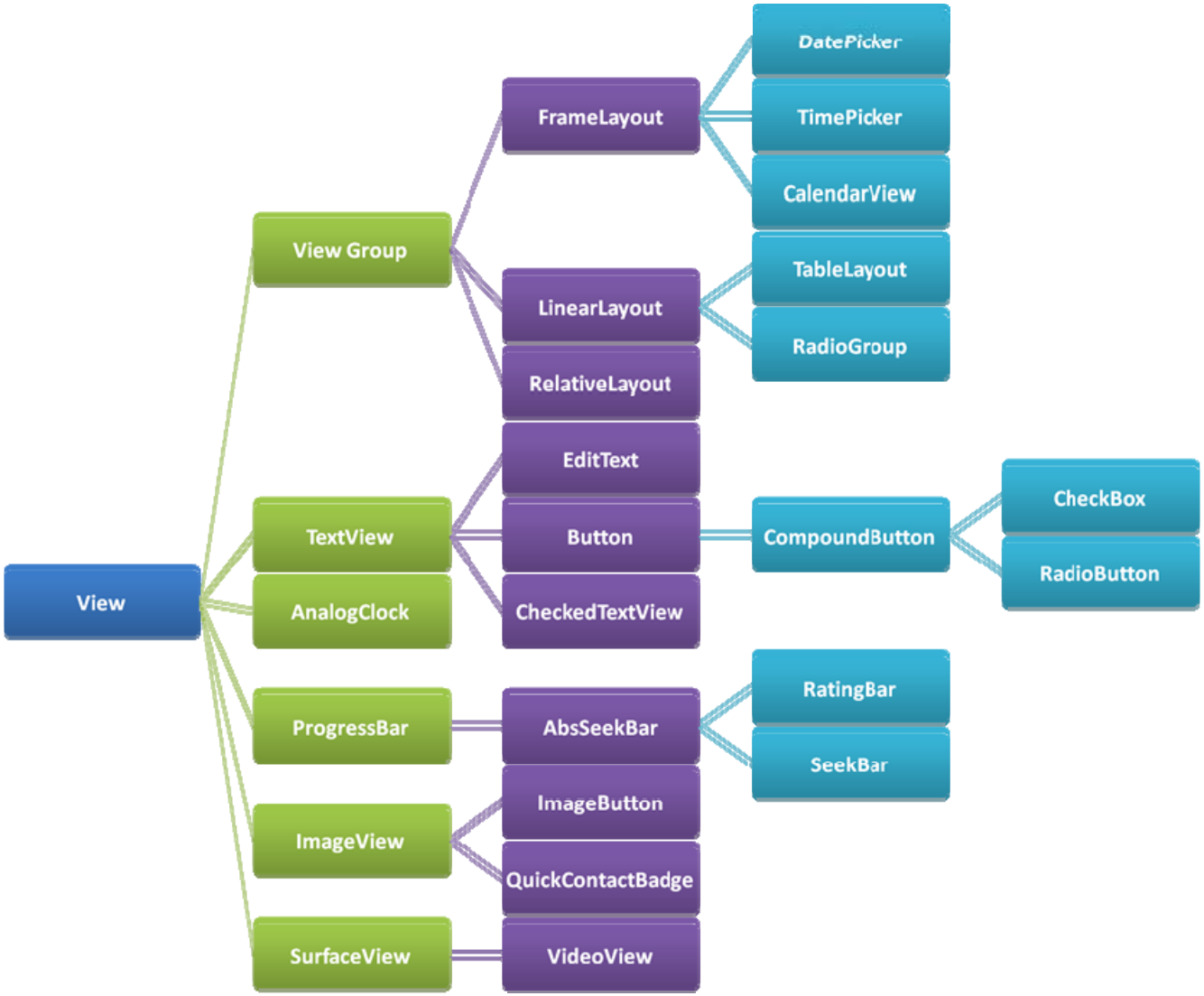
Android

IHM



Une vue est élément affichable de l'interface utilisateur (classe de base android.view.View).

Une vue de type ViewGroup peut contenir d'autres vues.

Hiérarchie des classes de l'interface utilisateur

Exemple d'un composant de type View

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

<Button
xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent"

android:layout_height="match_parent"
    android:text="Un bouton" >
</Button>
```

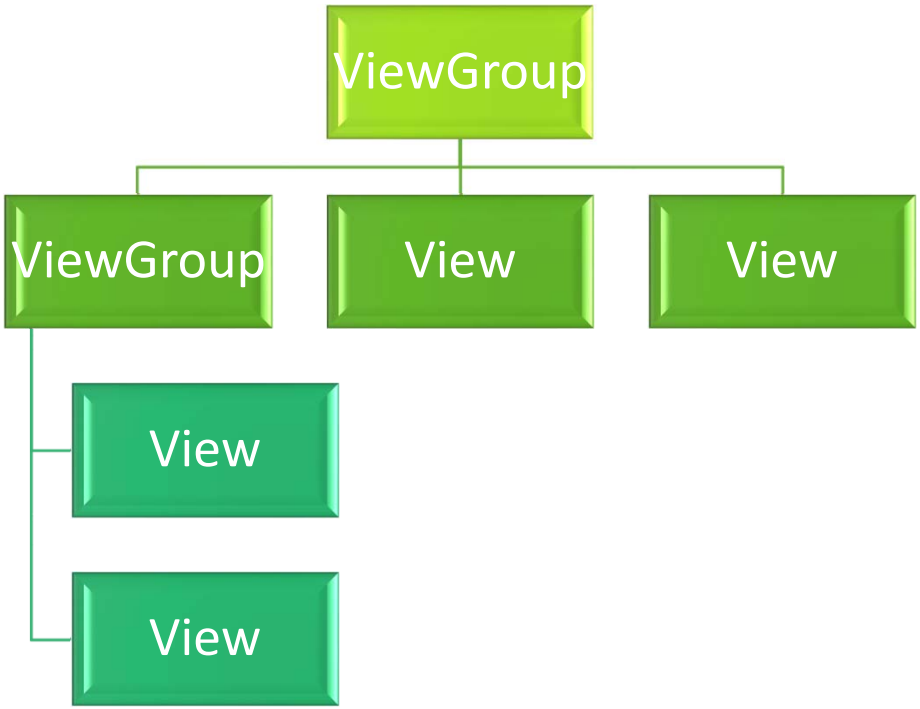
L'élément racine d'une vue doit déclarer le namespace « http://schemas.android.com/apk/res/android »

Valeurs possibles pour les attributs android:layout_width et android:layout_height:

- *match_parent* : l'élément remplit tout l'élément parent.
- *wrap_content* :le composant est dimensionné selon son contenu.

Id: l'id permet d'identifier la vue d'une manière unique dans l'arborescence. l'id est défini dans le document xml comme une chaîne, mais il est référencé dans le code java par un entier généré automatiquement.

Les vues



Une vue est un fichier xml enregistré dans le dossier res\Layout et qui contient une arborescence de composants graphique dont la racine est généralement un élément de type ViewGroup.

Le nom du fichier xml, par example main.xml permet de retrouver le layout dans le code java au travers de **R.layout.main**.

Pour afficher une vue à partir de l'activité:

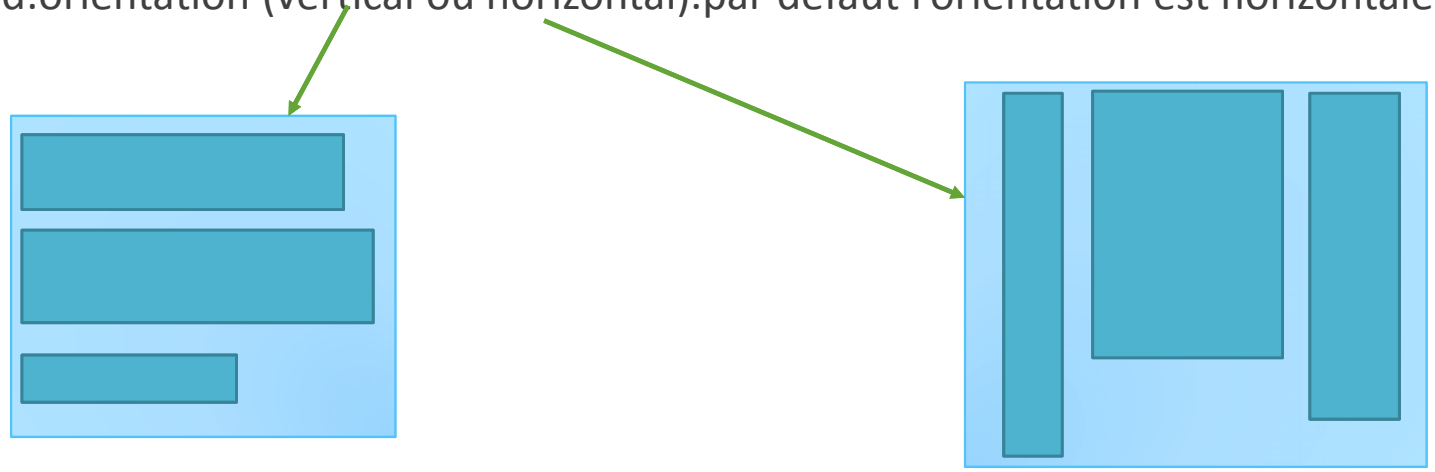
```
setContentView(R.layout.main);
```

Les gestionnaires de disposition

Un gestionnaire de disposition définit la structure visuelle d'une interface utilisateur.

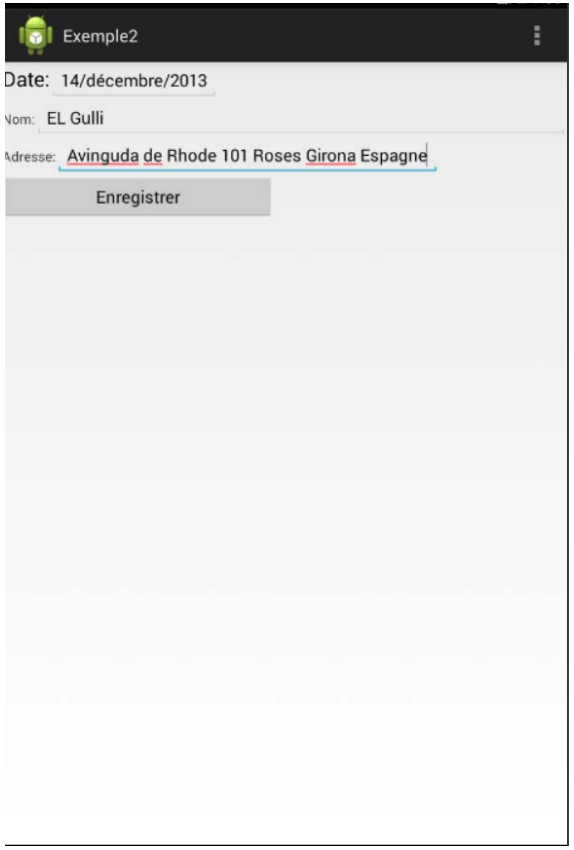
LinearLayout

Dispose les éléments de gauche à droite ou du haut vers le bas selon la valeur de la propriété android:orientation (vertical ou horizontal).par défaut l'orientation est horizontale.



Exemple

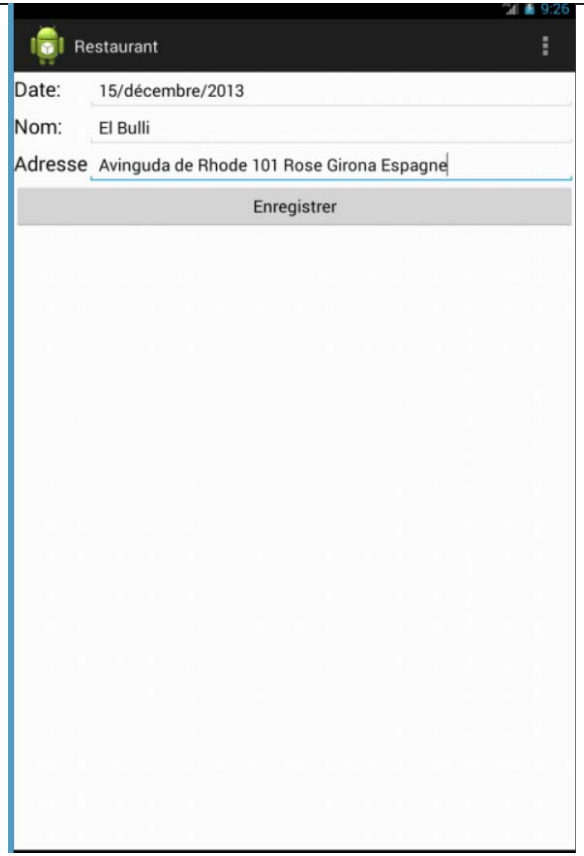
```
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Date:"/>
        <EditText android:id="@+id/txtDate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>
    <LinearLayout android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Nom:" />
        <EditText
            android:id="@+id/txtNom"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Adresse:" />
        <EditText
            android:id="@+id/txtAdresse"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>
      <Button  android:id="@+id/bEnregistrer"
        android:layout_width="289dp"
        android:layout_height="wrap_content"
        android:text="Enregistrer" />
</LinearLayout>
```

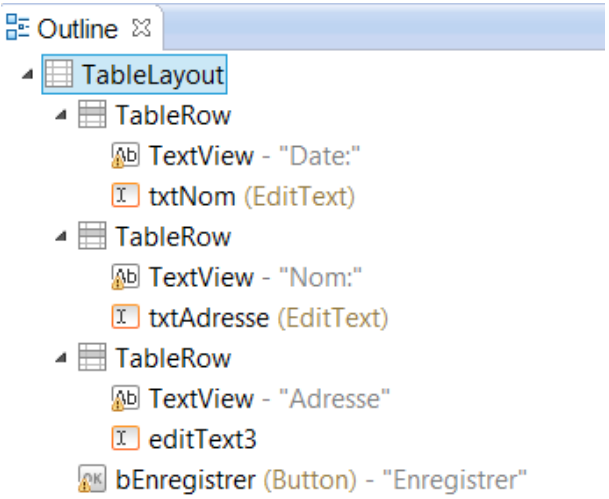


- LinearLayout: @+id/llv1
  - LinearLayout
    - TextView: @+id/textView1
    - EditText: @+id/txtNom
  - LinearLayout
    - TextView: @+id/textView2
    - EditText: @+id/txtAdresse
  - LinearLayout
    - TextView: @+id/textView3
    - EditText: @+id/editText3
  - Button: @+id/bEnregistrer

TableLayout

- Un TableLayout est composé d'éléments de type TableRow, il dispose les vues dans des lignes et des colonnes.
- Le nombre de colonnes est déterminé par la ligne qui contient le plus grand nombre de cellules.
- La largeur d'une colonne est définie par la cellule la plus large de cette colonne.
- android:stretchColumns=*"1":* La colonne 1 (EditText) occupe la largeur restante dans la ligne.



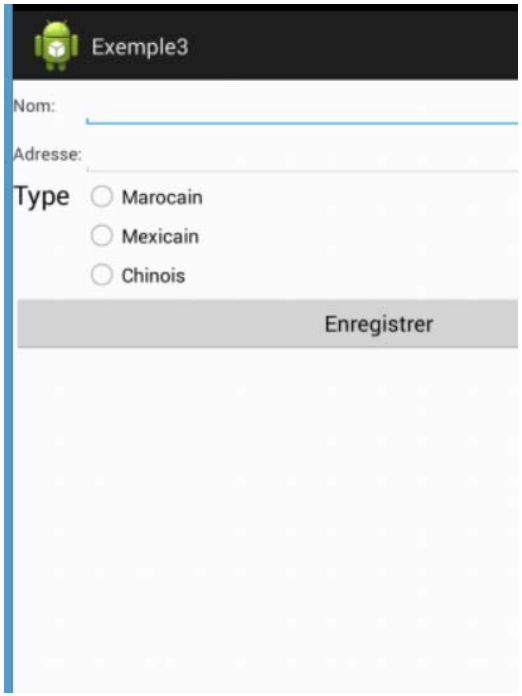


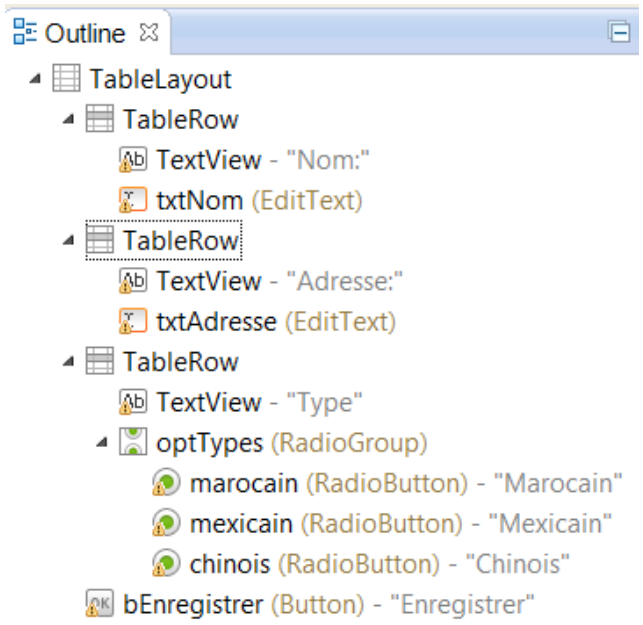

```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1" >
    <TableRow>
        <TextView android:text="Date:" />
        <EditText android:id="@+id/txtNom" />
    </TableRow>
    <TableRow>
        <TextView android:text="Nom:" />
        <EditText android:id="@+id/txtAdresse" />
    </TableRow>
    <TableRow>
        <TextView android:text="Adresse" />
        <EditText android:id="@+id/editText3" />
    </TableRow>
    <Button android:id="@+id/bEnregistrer"
        android:text="Enregistrer" />
</TableLayout>
```

L'activité

```
protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);



/* Affichage de la date */

DateFormat formatDate = new SimpleDateFormat("dd/MMMM/yyyy",Locale.FRENCH);

String date = formatDate.format(Calendar.getInstance().getTime());

EditText txtDate = (EditText) findViewById(R.id.txtNom);

txtDate.setText(date);



/* Création et initialisation du bouton Enregistrer */

Button cmdEnregistrer = (Button) findViewById(R.id.bEnregistrer);

cmdEnregistrer.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {


restaurant = new Restaurant();

EditText txtNom = (EditText) findViewById(R.id.txtAdresse);


EditText txtAdresse = (EditText) findViewById(R.id.editText3);


restaurant.setNom(txtNom.getText().toString());

restaurant.setAdresse(txtAdresse.getText().toString());


}

});

}
```

Ajout d'un groupe de boutons radios






```
<RadioGroup android:id="@+id/optTypes" >
            <RadioButton
                android:id="@+id/marocain"
                android:text="Marocain" />
            <RadioButton
                android:id="@+id/mexicain"
                android:text="Mexicain" />

            <RadioButton
                android:id="@+id/chinois"
                android:text="Chinois" />
</RadioGroup>
```

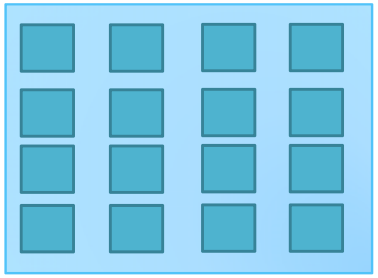
Ajout d'un groupe de boutons radios

```
//Récupéré le type sélectionné.

RadioGroup types=
(RadioGroup)findViewById(R.id.optTypes);

    switch (types.getCheckedRadioButtonId())
    {

    case
    R.id.marocain:restaurant.setType("Marocain");
    break;
    case
    R.id.mexicain:restaurant.setType("Mexicain");
    break;
    case
    R.id.chinois:restaurant.setType("Chinois");
    break;

    }
```

La méthode getCheckedRadioButtonId de RadioGroup retourne l'Id du bouton radio sélectionné.

Layout dynamiques

GridView



ListView

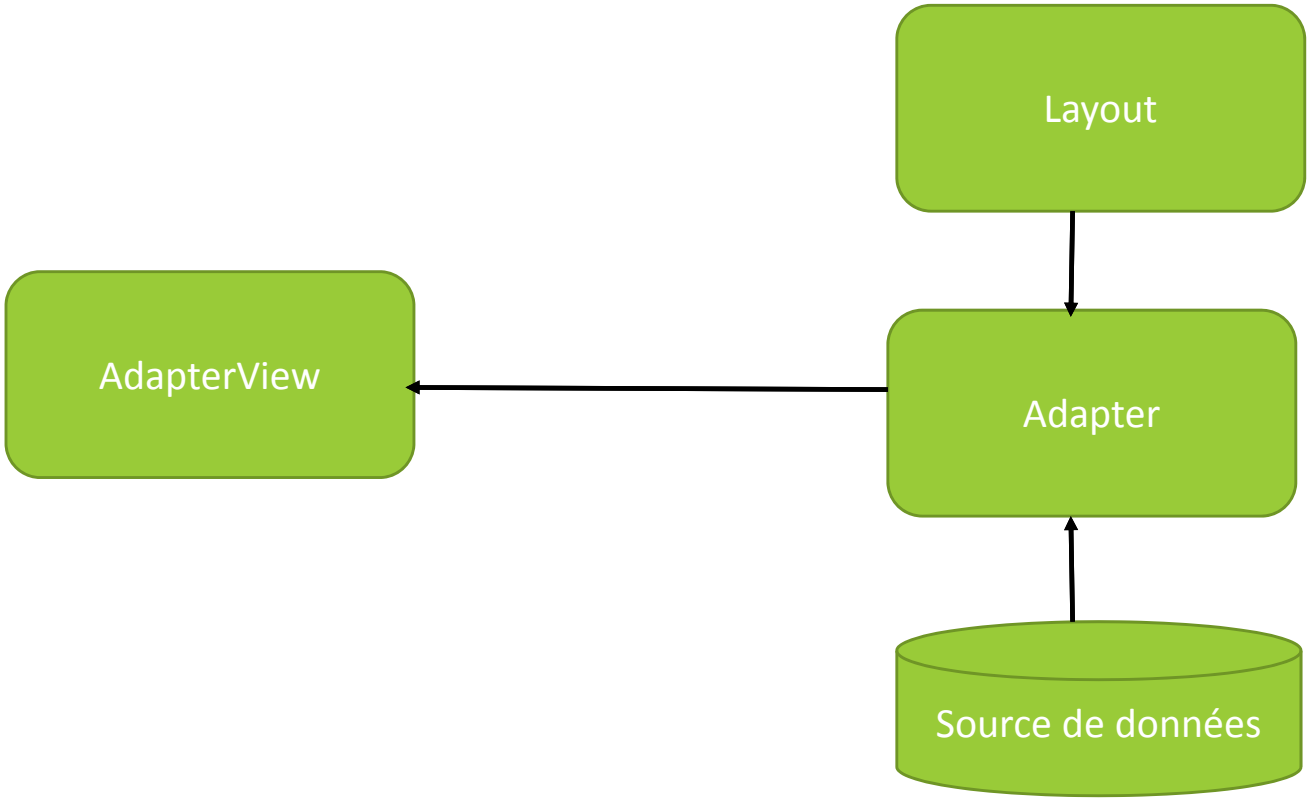
Les layouts dynamiques dérivent de AdapterView et affichent un ensemble d'éléments.

AdapterView<T extends android.widget.Adapter> est une classe abstraite

Exemples d'implémentations de AdapterView:

- ListView
- GridView
- WebView
- Spinner

Fonctionnement d'un AdapterView

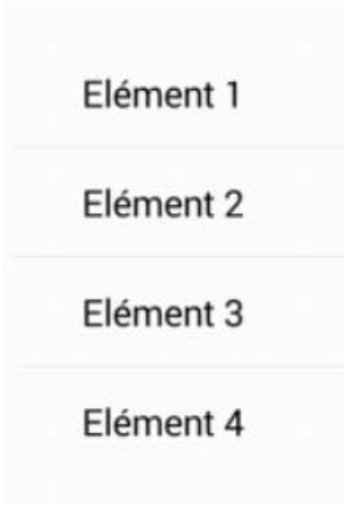


Le Layout est utilisé par un objet Adapter pour créer une vue pour chaque élément obtenu à partir de la source de données dans un élément de type AdapterView, Android fournit plusieurs implémentations de la classe Adapter:

- ArrayAdapter
- SimpleCurser

Exemple ListView

```
// Source de données
String[] source = { "Elément 1", "Elément 2", "Elément 3", "Elément 4" };
// Adapter
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,// Contexte
android.R.layout.simple_expandable_list_item_1,// id du layout
  // prédéfini à utilisé pour afficher la liste
android.R.id.text1,// De type TextView
source);
// Récupérer l'id de la liste
ListView lv = (ListView) findViewById(R.id.listView1);
// Lier la liste à adapter
lv.setAdapter(adapter);
```



L'événement Clic

```
liste.setOnItemClickListener(new OnItemClickListener() {


@Override
public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
/* parent: ListView Concernée par le clic
 * View: Le composant View à l'origine de l'événement
 * position: indice de l'élément sélectionné dans l'adapter
 * id: id de la ligne concernée par le clic
 * */


}
});
```

L'événement Sélection

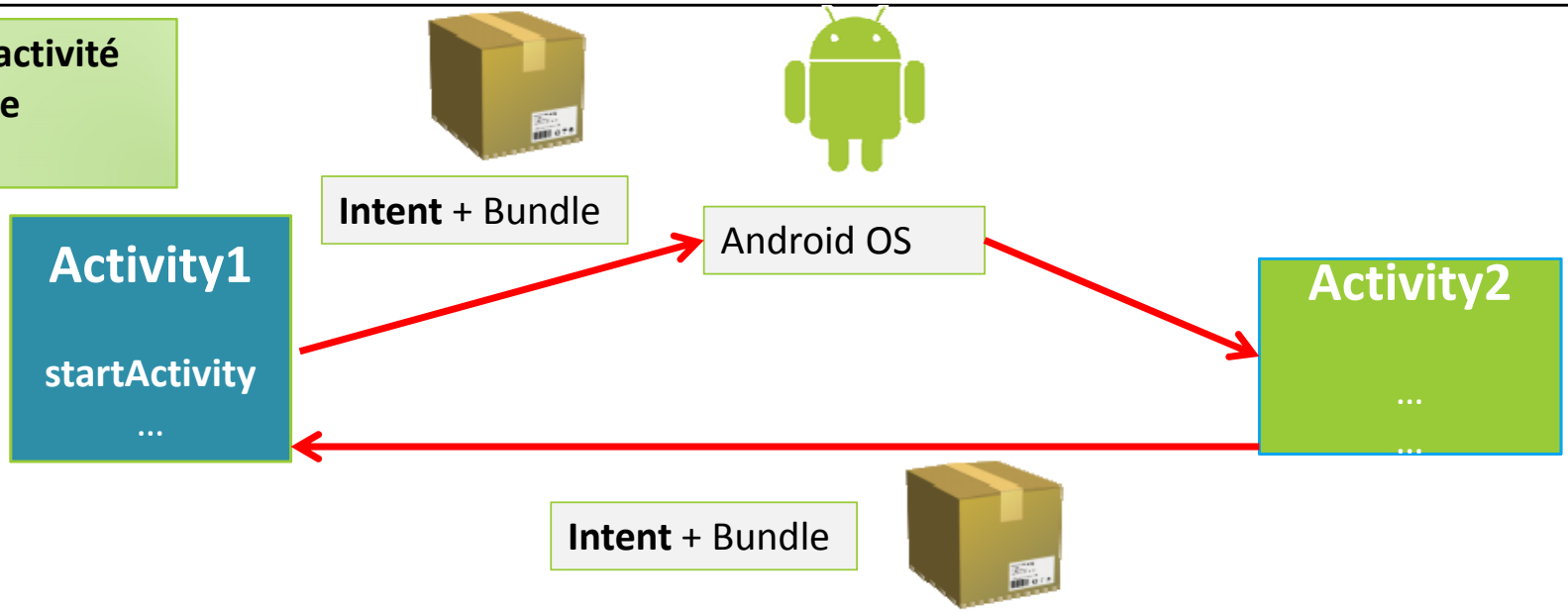
```
list.setOnItemSelectedListener(new OnItemSelectedListener() {

@Override
public void onItemClick(AdapterView<?> parent, View view,
int index, long id) {
…

}

@Override
public void onNothingSelected(AdapterView<?> parent) {
// déclenché si plus aucun élément n'est sélectionné (Adapter vide par exemple)

}
});
```

Passer d'une activité à une autre

Un objet de type Intent est utilisé pour passer d'une activité à une autre.

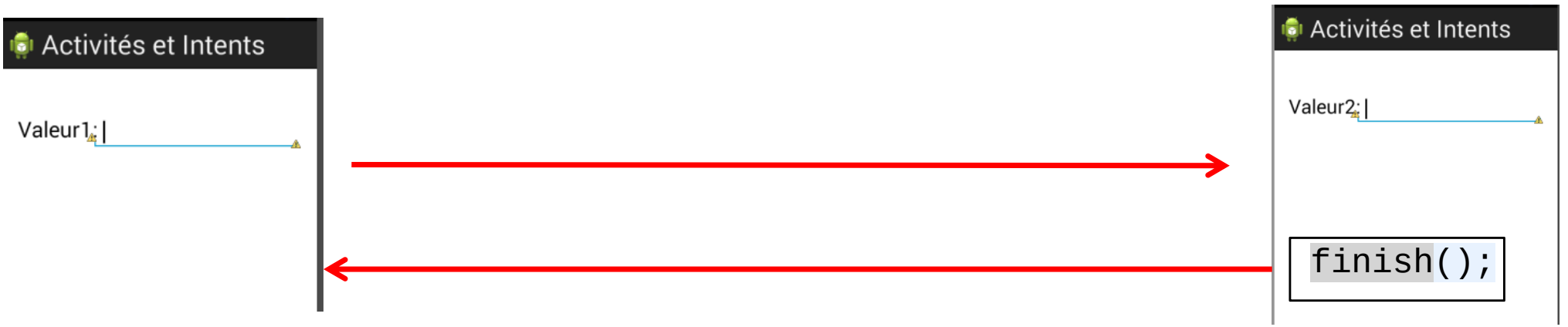
Un objet Intent (ou intention) est créé dans l'activité de départ

Démarre une activité sans attente de résultat



Scénario 1 : Afficher Activité sans attente de résultat et sans en envoi de données.

```
Intent intentAct2=new Intent (this,// Contexte
Activite2.class // Activité à démarrer
);
startActivity(intentAct2);
```



Scénario 2 : Afficher Activité sans attente de résultat et avec envoi de données.

```
Intent intentAct2 = new Intent(this, Activite2.class);
intentAct2.putExtra("valeur", 5);//L'objet Intent possède
/* un Bundle nommé extra (un Bundle est une structure de dictionnaire */
startActivity(intentAct2);
```

Dans l'événement onCreate

```
int extra = (int) getIntent().getIntExtra("valeur", 0);
 //le deuxième paramètre est la valeur par défaut
/*Pour un objet la classe doit être serialisable
extra=getIntent().getSerializableExtra("restaurant");
 */EditText t = (EditText) findViewById(R.id.editText1);
t.setText(String.valueOf(extra));
```

Scénario 3 : Activité avec attente de résultat

Pour démarrer l'activité 2

```
startActivityForResult(intentAct2, 1);
```

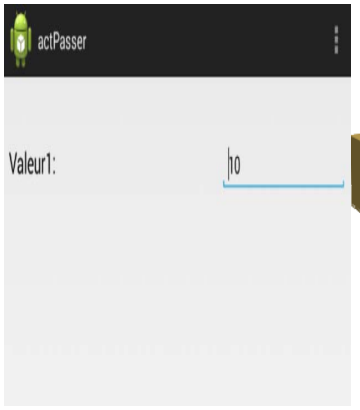
1 permet d'identifier la requête.

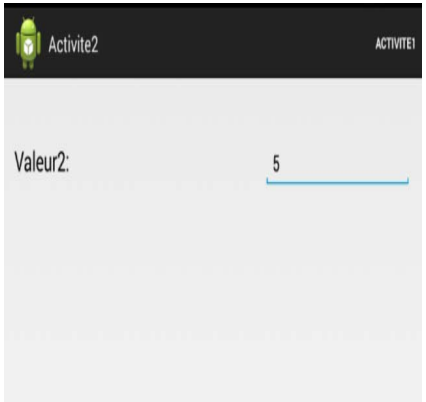
Réception du résultat dans activité1

```
protected void onActivityResult(int requestCode,
int resultCode, Intent data) {

if (requestCode==1){int
val=data.getIntExtra("valeur", 0);

TextView t=(TextView)findViewById(R.id.editText1);

t.setText(String.valueOf(val));}

super.onActivityResult(requestCode, resultCode,
data);}
```

Retour du résultat

```
Intent intentionRetour = new
Intent();

intentionRetour.putExtra("valeur",
10);

setResult(RESULT_OK,
intentionRetour);

finish();
```

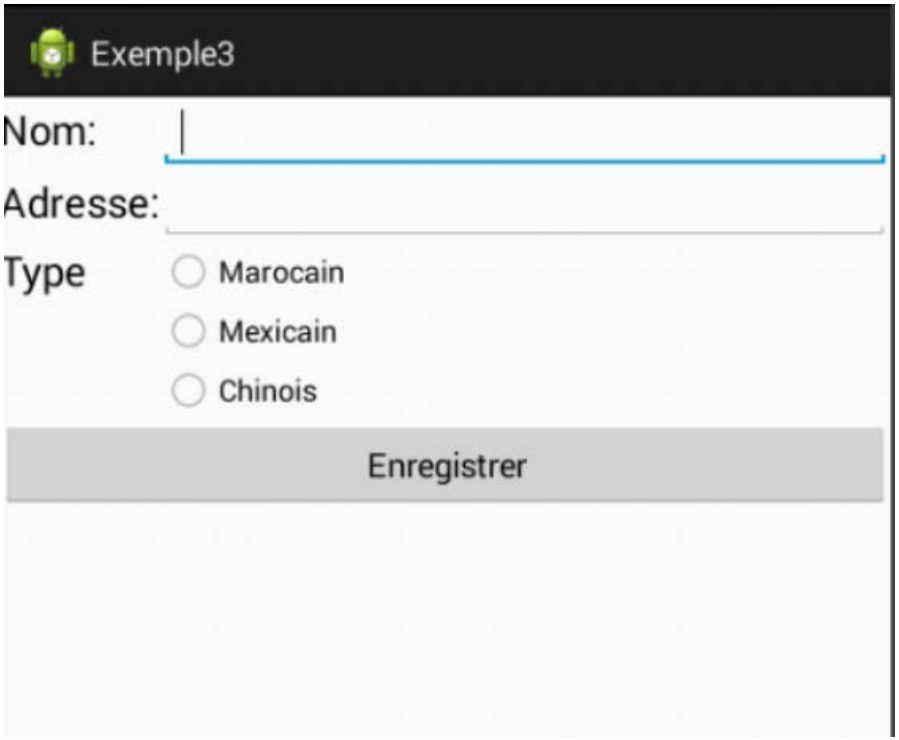




TP: Exemple3

Etape1: ajout d'un groupe de boutons radios

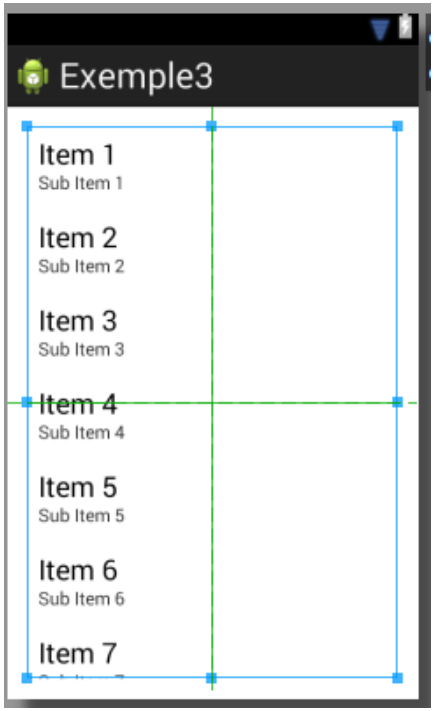
- Ajouter le champ type dans la classe Restaurant
- Ajouter un constructeur à trois paramètres dans la classe
- Ajouter le groupe de boutons radios
- Mettre ajouter l'enregistrement d'un restaurant en tenant compte du type de restaurant

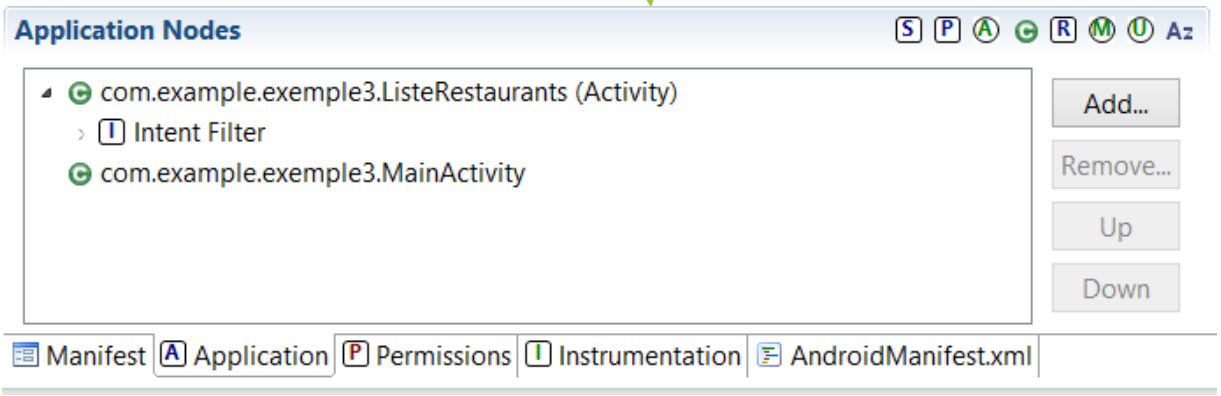


Etape 2: Ajouter une activité pour afficher la liste des restaurants

- Ajouter une activité nommée ListeRestaurants
- Ajouter une ListView dans le layout
- Dans le fichier AndroidManifest.xml définir l'activité ListeRestaurants comme activité de démarrage.

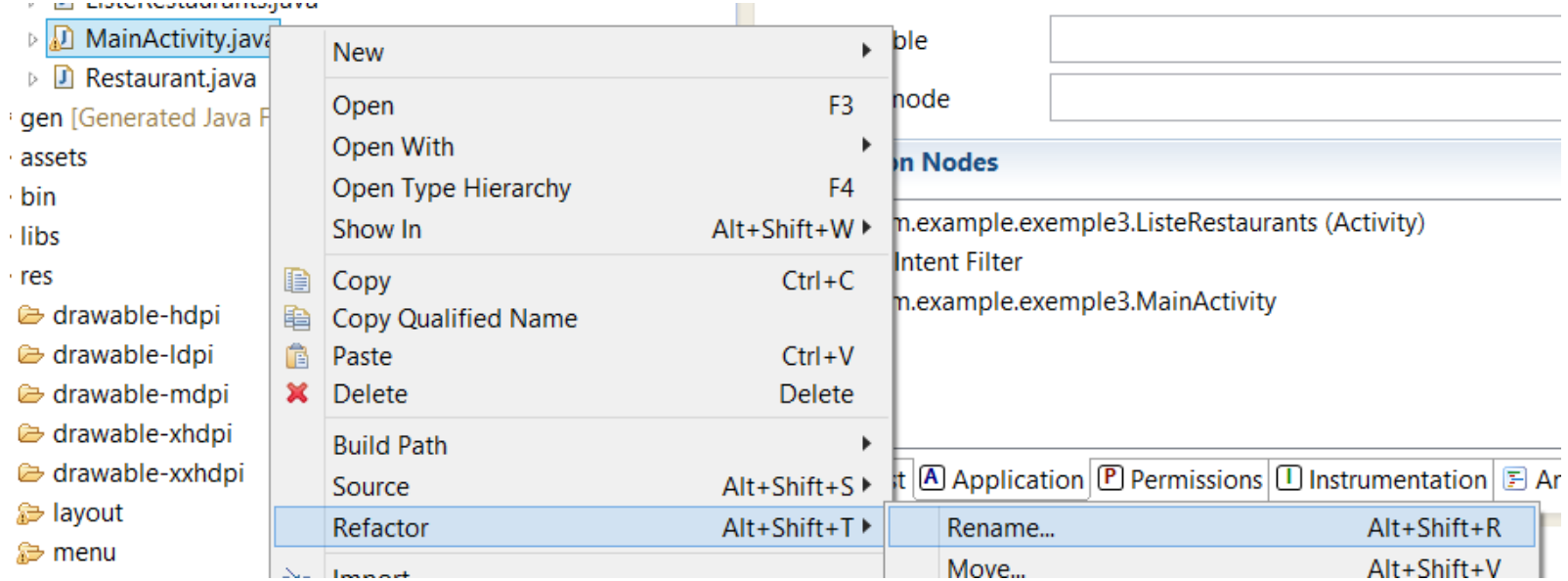
```
<ListView
        android:id="@+id/lvRestau"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
>
    </ListView>
```





Etape 3: Renommer « ActivityMain »

- Renommer l'activité (Refactor/Rename), nouveau nom: ModifierRestaurant
- Renommer le layout (Refactor/Rename), nouveau nom:modifier_restaurant.xml
- Renommer le menu , nouveau nom: modifier_restaurant.xml



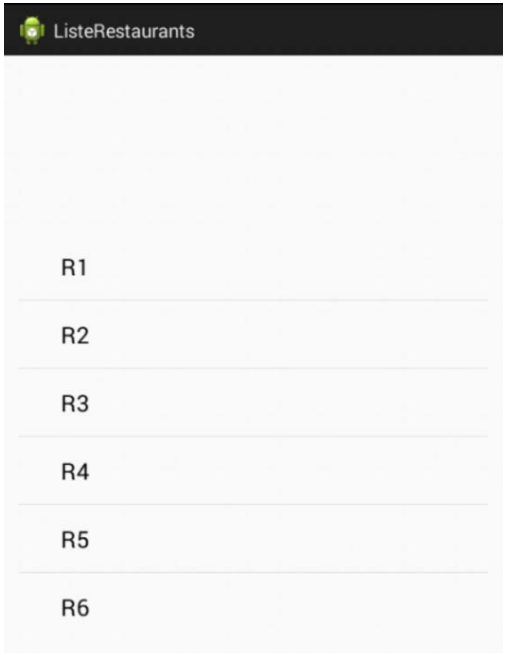
Etape 4: Remplir la liste

- Dans la classe « ListeRestaurants » ajouter les attributs privés

```
ListView lv ;// Liste des restaurants
List<Restaurant> restaurants=new
ArrayList<Restaurant>();
```

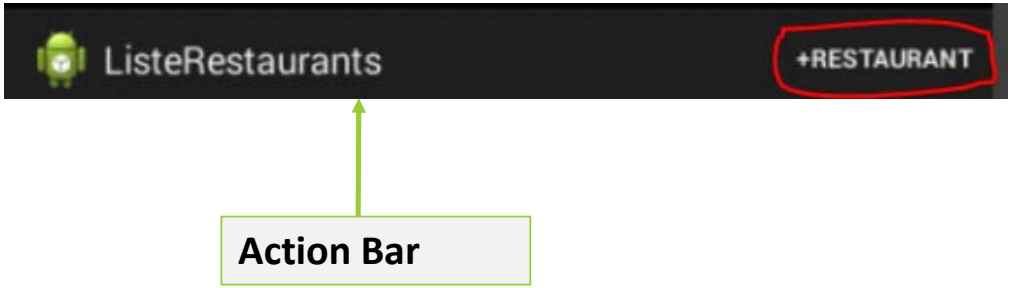
- dans l'événement onCreate ajouter le code pour remplir la liste

```
restaurants.add(new Restaurant("R1", "Adresse 1",
"Marocain"));
restaurants.add(new Restaurant("R2", "Adresse 2",
"Marocain"));
restaurants.add(new Restaurant("R3", "Adresse 3",
"Mexicain"));
restaurants.add(new Restaurant("R4", "Adresse 4",
"Mexicain"));
restaurants.add(new Restaurant("R5", "Adresse 5",
"Chinois"));
restaurants.add(new Restaurant("R6", "Adresse 6",
"Chinois"));
ArrayList<String> source=new ArrayList<String>();
// Création de la source de données
for (Restaurant r:restaurants)
source.add(r.getNom());
// Création de l'adapter
ArrayAdapter<String> adapter=new
ArrayAdapter<String>(this,
android.R.layout.simple_expandable_list_item_1,
android.R.id.text1, source);
// Lier lv à adapter
lv= (ListView)findViewById(R.id.lvRestau);
lv.setAdapter(adapter);
```



Etape 5: Ajouter un élément de menu

- Dans le fichier de ressources res/menu/liste_restaurants.xml, modifier l'élément de menu existant (ou ajouter un nouveau):
  - id:*@+id/ajouter*
  - *Show as Action: cocher les cases « withText » et « alwyas » (l'élément de menu s'affichera dans la barre d'action*
  - *Title: +Restaurant*



- L'affichage du menu dans la barre d'action est réalisé dans l'activité, à l'aide de la méthode (générée automatiquement lors de la création de l'activité

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it
is present.
getMenuInflater().inflate(R.menu.liste_restaurants, menu);
return true;
}
```

Etape 6: Implémenter l'ajout d'un restaurant

Etape 6.1: afficher l'activité « ModifierRestaurant »

- Lors du clic sur l'élément de menu « +Restaurant », on doit afficher l'activité « ModifierRestaurant ».
- Le clic sur un élément de menu déclenche l'exécution de la méthode « onOptionsItemSelected », qui fournit en argument l'objet MenuItem concerné par le clic.
- Pour ajouter la méthode onOptionsItemSelected à l'aide de l'assistant :
  - Clic droit sur le nom de la classe : Source/ « Override/implement Method » et cocher la méthode dans la liste.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
Intent intent = new Intent(this, ModifierRestaurant.class);
/*Le deuxième paramètre est une constante qui identifie
 * le type de la requête, il est préférable d'utiliser
 * une valeur qui ne soit pas déjà utilisé par les
 * constantes prédéfinies dans Android.
 * RESTO_AJOUT est une constante définie dans
 * la classe :
 * public static final int RESTO_AJOUT = 555;
 */
startActivityForResult(intent, RESTO_AJOUT);
return super.onOptionsItemSelected(item);
}
```

Etape 6.2: Compléter le bouton « Enregistrer de l'activité « ModifierRestaurant »

- Le clic sur le bouton « Enregistrer » enregistre les données saisies dans les vues dans un objet restaurant et retourne le retourne à l'activité appelante.
- La classe Restaurant doit implémenter l'interface Serializable.
- Dans l'événement clic sur le bouton « Enregistrer », ajouter le code pour retourner l'objet restaurant à l'acitivité « ListeRestaurants ».

```
Intent intentionRetour=new Intent();
intentionRetour.putExtra("restaurant", restaurant);
setResult(RESULT_OK,intentionRetour);
finish();}
```

Etape 6.3: Afficher dans la liste le nom et le type du restaurant

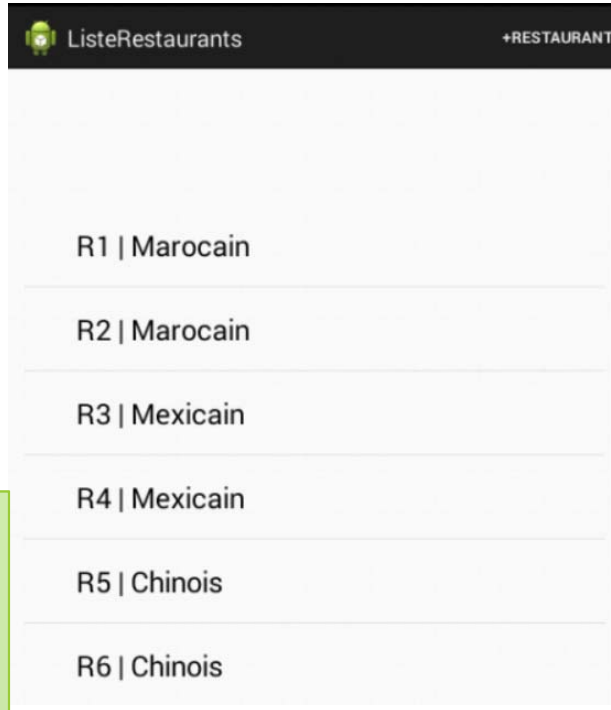
- Pour afficher un élément dans la liste, ArrayAdapter appelle la méthode toString().
- Implémenter la méthode toString() dans la classe Restaurant:

```
@Override
public String toString() {
return String.format("%s | %s ",nom, type);}
```

- Changer la variable locale « adapter » déclarée dans la méthode « onCreate » en un champ de la classe « ListeRestaurants »: sélectionnez adapter puis cliquez avec le bouton droit refactor/Convert Local variable to Fied.
- Dans la méthode onCreate, supprimer la déclaration et le remplissage du tableau source, il ne sera plus utilisé.

- Dans l'événement clic sur le bouton « Enregistrer », ajouter le code pour retourner l'objet restaurant à l'activité « ListeRestaurants ».
- Utiliser « restaurants » comme source de données de « adapter »

```
adapter = new ArrayAdapter<Restaurant>(this,
android.R.layout.simple_expandable_list_item_1,
android.R.id.text1, restaurants);
```



Etape 6.3: Récupérer et afficher l'objet « restaurant »

- Dans l'activité « ListeRestaurants » implémenter la méthode onActivityResult, pour récupérer le nouveau restaurant.

```
@Override
protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
Serializable extra = data.getSerializableExtra("restaurant");
if (extra != null) {
if (requestCode == RESTO_AJOUT)
adapter.add((Restaurant) extra);

}

super.onActivityResult(requestCode, resultCode, data);
}
```

Etape 7: Modifier un restaurant

- Dans la méthode onCreate de l'activité « ListeRestaurants » définir l'événement ItemClickListener de la liste lv

```
lv.setOnItemClickListener(new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> adapter, View view, int index,
long id) {
/*pour le contexte on ne pas utiliser this qui référence la classe anonyme
 * il doit être récupéré à partir de view.getContext()
*/
Intent modifierRestoIntent = new Intent(view.getContext(), ModifierRestaurant.class);
modifierRestoIntent.putExtra("restaurant", restaurants.get(index));
//iCourant doit être déclaré comme un champ de la classe, il sauvegarde l'indice de l'élément
sélectonné.
iCourant=index;
//RESTO_MODIFIE=556
startActivityForResult(modifierRestoIntent, RESTO_MODIFIE);
}
});
```

- Dans la méthode onCreate de l'activité « ModifierRestaurant », récupérer et afficher le restaurant

```
Serializable extra= getIntent().getSerializableExtra("restaurant");
if (extra!=null){
restaurant=(Restaurant)extra;
EditText txtNom = (EditText) findViewById(R.id.txtNom);
EditText txtAdresse = (EditText) findViewById(R.id.txtAdresse);
RadioGroup optTypes=(RadioGroup)findViewById(R.id.optTypes);
txtNom.setText(restaurant.getNom());
txtAdresse.setText(restaurant.getAdresse());
if (restaurant.getType().equals("Marocain"))
optTypes.check(R.id.marocain);
if (restaurant.getType().equals("Mexicain"))
optTypes.check(R.id.mexicain);
if (restaurant.getType().equals("Chinois"))
optTypes.check(R.id.chinois);
}
```

- Dans la méthode onActivityResult de l'activité « ListeRestaurants », il faut récupérer le restaurant modifié et l'enregistrer à la position iCourant

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
…
if (extra != null) {
…
// Modifier l'élément à la position iCourant
if (iCourant != -1) {
restaurants.set(iCourant, (Restaurant) extra);
adapter.notifyDataSetChanged();
// Réinitialiser iCourant
iCourant = -1;
}
}
…
}
```

Etape 7: Implémenter le bouton « Annuler » dans l'activité Modifier « Restaurant »

- Dans le layout modifier_restaurant.xml ajouter un bouton « annuler ».

```
<Button
        android:id="@+id/bAnnuler"
        android:text="Annuler" />
```

- Dans la méthode onCreate de l'activité « ModifierRestaurant », ajouter l'événement clic du bouton bAnnuler

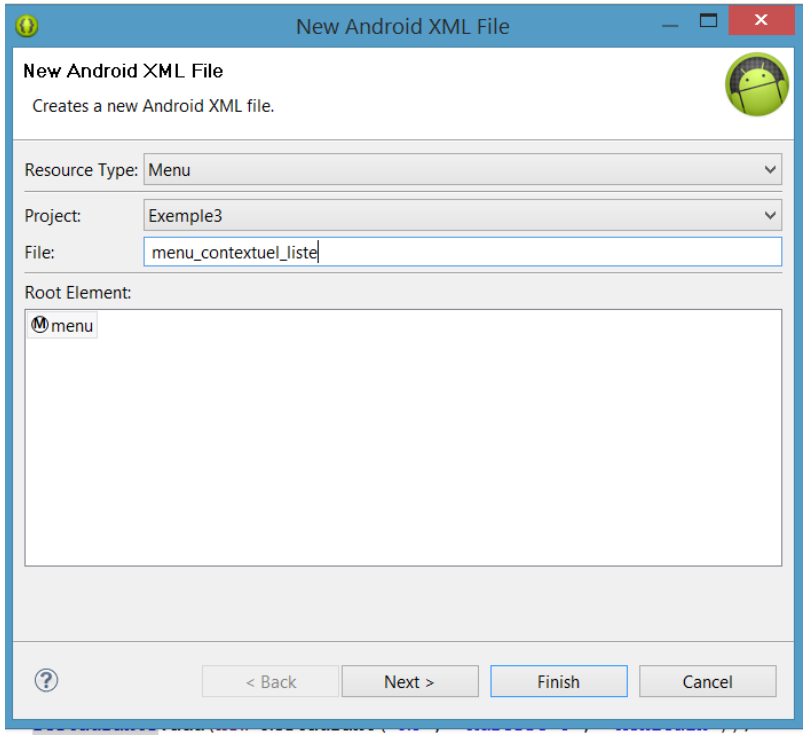
```
Button cmdAnnuler=(Button)findViewById(R.id.bAnnuler);
cmdAnnuler.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View v) {


setResult(RESULT_CANCELED, new Intent());
finish();

}
});
```

- Dans la méthode onActiviyResult de l'activité « ListeRestaurants », ajouter le code suivant pour traiter l'action « Annuler »:

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
if (resultCode==RESULT_CANCELED)
return;
```

Etape 8: Suppression d'un élément de la liste à partir d'un menu contextuel

- Ajouter une ressource de type menu (clic droit sur le dossier res/menu et « New/Android Xml File »



- Ajouter l'élément de menu suivant:
  - id: supprimer
  - Titre: Supprimer
- Implémenter le menu contextuel (la méthode **onCreateContextMenu**) dans l'activité ListeRestaurants

```
@Override
public void onCreateContextMenu(ContextMenu
menu, View v,
ContextMenuInfo menuInfo) {
super.onCreateContextMenu(menu, v, menuInfo);
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.menu_contextuel_liste,
menu);
}
```

- Dans la méthode onCreate enregistrer la liste lv pour supporter les menus contextuels

```
registerForContextMenu(lv);
```

- Implémenter la méthode onContextItemSelected() qui est appelée lors de la sélection d'une action dans le menu contextuel.

```
@Override
public boolean onContextItemSelected(MenuItem
item) {
AdapterContextMenuInfo
info=(AdapterContextMenuInfo)item.getMenuInfo();
restaurants.remove( info.position);
adapter.notifyDataSetChanged();
return super.onContextItemSelected(item);

}
```