le paramètre gravity

- Dans certains cas, vous serez amené à positionner des vues plutôt que de les aligner.
- Pour se faire, il faudra utiliser le paramètre **gravity**,
- le paramètre **gravity** spécifie l'attrait d'un composant vers un coin de l'écran.

le paramètre gravity

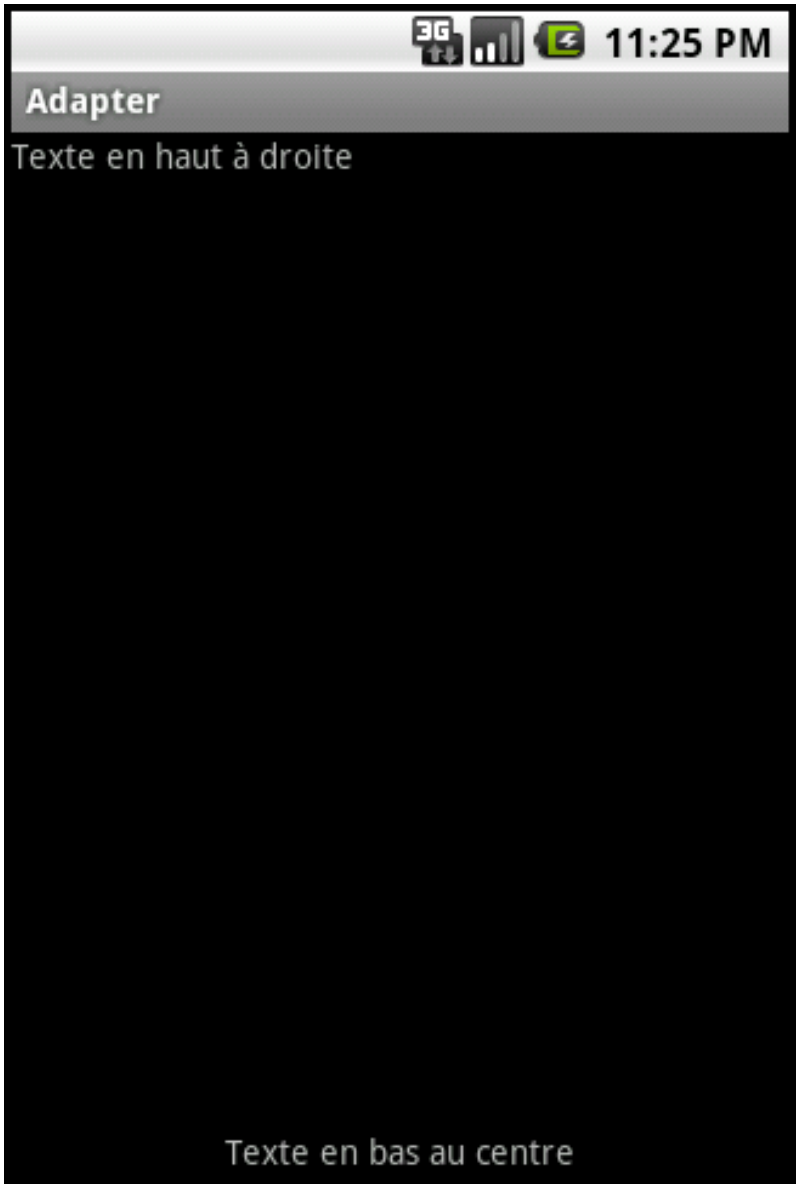
- Nous allons maintenant modifier **la gravité** de deux éléments **TextView** dans la définition de l'interface afin de les aligner en **haut** et en **bas** de notre gabarit.

- Vous noterez l'utilisation du caractère | pour spécifier une combinaison de valeurs et placer vos éléments au plus proche de vos besoins.

le paramètre gravity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical"
  >
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/monText"
    android:text="Texte en haut à droite"
    android:gravity="top|right"
    />
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="bottom|center_horizontal"
    android:id="@+id/monText2"
    android:text="Texte en bas au centre"
    />
</LinearLayout>
```

le paramètre gravity



le paramètre gravity

- Dans cet exemple, la gravité top | right (haut | droite) est appliquée sur l'élément TextView (*texte en haut à droite*) qui occupe toute la largeur grâce à la valeur fill_parent du paramètre layout_width. Ce paramétrage indique au texte de se positionner le plus en haut et à droite de l'espace qu'il occupe.

- Si à la place de la valeur fill_parent nous avions spécifié wrap_content, le texte aurait été aligné visuellement à gauche car la place qu'occuperait l'élément TextView serait limitée à son contenu.

le paramètre gravity

- Pour aligner en bas le deuxième TextView, nous avons dû paramétrer la propriété layout_width et layout_height avec la valeur fill_parent et spécifier la valeur bottom | center_horizontal (bas | centre horizontal) à la propriété gravity.

Intégrer une image dans votre interface

- Pour ajouter une image dans votre interface, utilisez la vue de type **ImageView**.

- Vous pouvez spécifier la source de l'image directement dans votre définition XML, via l'attribut **src**,

    - ou alors utiliser la méthode **setImageResource** en passant **l'identifiant** en paramètre.

- Pour l'exemple suivant, nous utilisons une image nommée **icon.png** qui se trouve déjà dans les ressources de votre projet (cette ressource est incluse dans le modèle de projet Android d'Eclipse et se trouve dans le dossier **/res/drawable**).

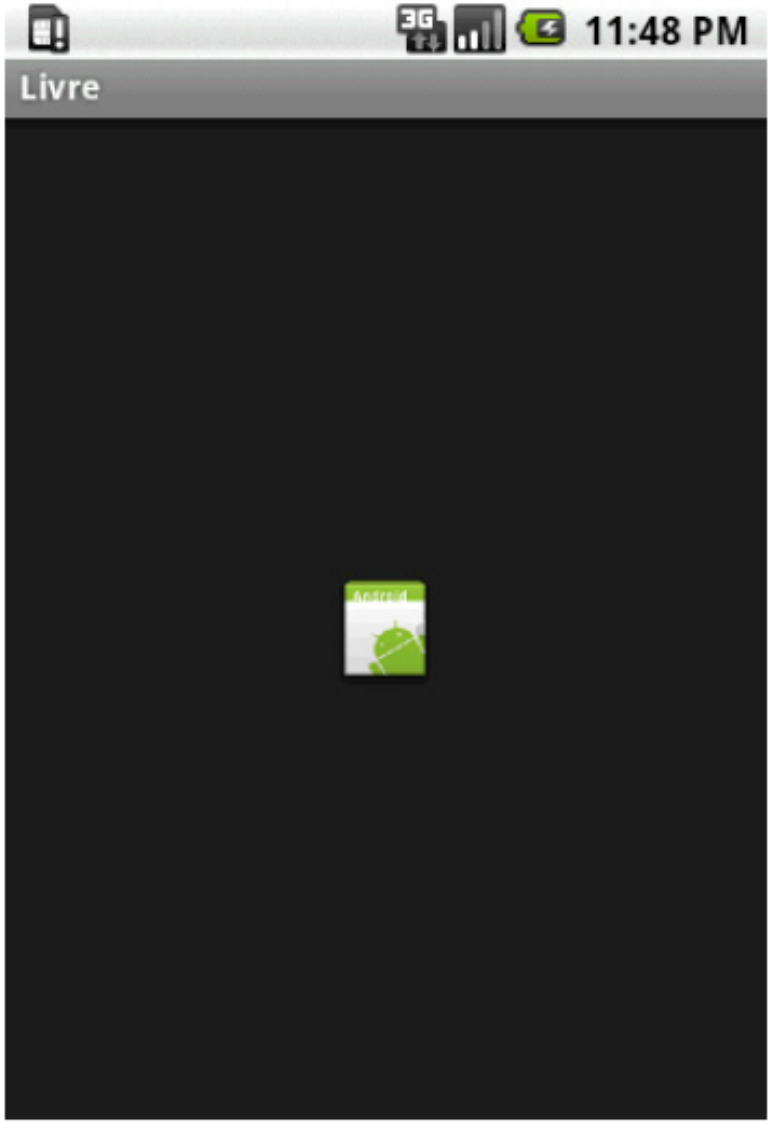
Intégrer une image dans votre interface

- Pour spécifier la taille d'une image dans la valeur de la propriété **src**, plusieurs options s'offrent à vous :
  - soit vous utilisez la dimension réelle de l'image **(wrap_content)**
  - ou la taille de l'écran **(fill_parent),**
  - soit vous spécifiez la taille exacte (exemple : **100 px, 25 dp,** etc).

Intégrer une vue image dans une interface

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_height="fill_parent"
   android:layout_width="fill_parent"
   android:gravity="center_vertical|center_horizontal"
  >
   <ImageView
      android:id="@+id/monImage"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/icon"
   >
   </ImageView>
</LinearLayout>
```

Intégrer une vue image dans une interface



Intégrer une vue image dans une interface

```
ImageView monImage = ...
monImage.setImageResource(R.drawable.icon);
```

Intégrer une boîte de saisie de texte

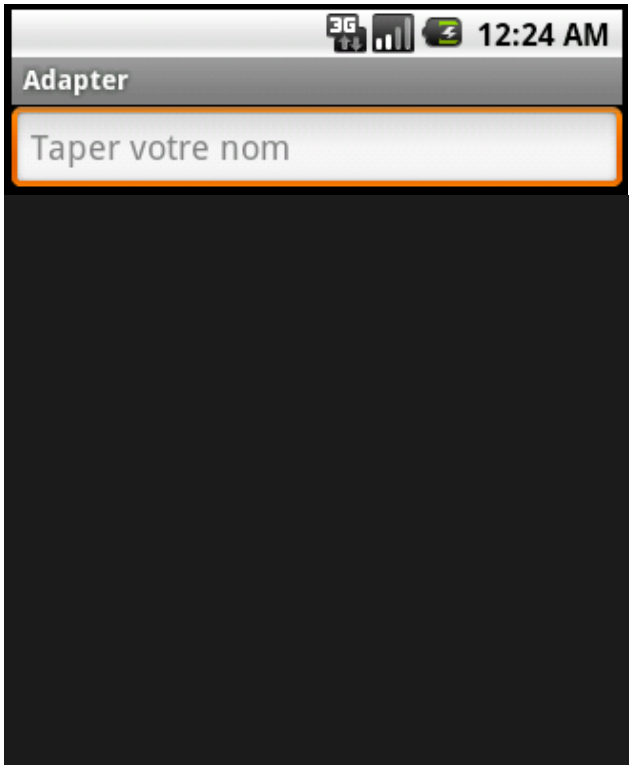
- Pour proposer à l'utilisateur de saisir du texte, vous pouvez utiliser la vue **EditText**.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <!--
       L'élément EditText permettra à
       l'utilisateur de saisir son nom.
    -->
    <EditText
       android:id="@+id/monEditText"
       android:layout_height="wrap_content"
       android:hint="Taper votre nom"
       android:layout_width="fill_parent">

    </EditText>
```

Ajout d'une vue de saisie de texte à l'interface utilisateur

- La propriété **hint** de l'élément **EditText** permet l'affichage en **fond** d'une aide.
- Tant qu'aucun texte n'est saisi dans la zone dédiée, cette aide apparaît **grisée** sur le fond.
- Cela peut vous éviter de mettre un texte de description avant ou au-dessus de chacune des zones de saisie et d'indiquer à l'utilisateur **l'objet de sa saisie.**

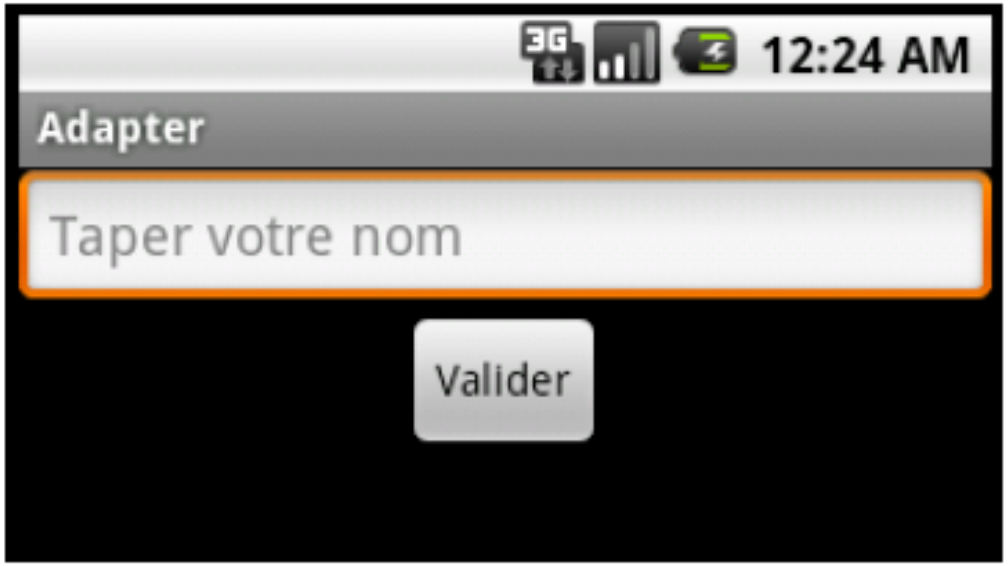
Ajout d'une vue de saisie de texte à l'interface utilisateur



Bouton de validation de saisie

```
    <!--
       Le Bouton qui permettra à l'utilisateur
       de valider sa saisie
    -->
    <Button
       android:id="@+id/monBouton"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Valider"
       android:layout_gravity="center_horizontal">
    </Button>
</LinearLayout>
```

Bouton de validation de saisie



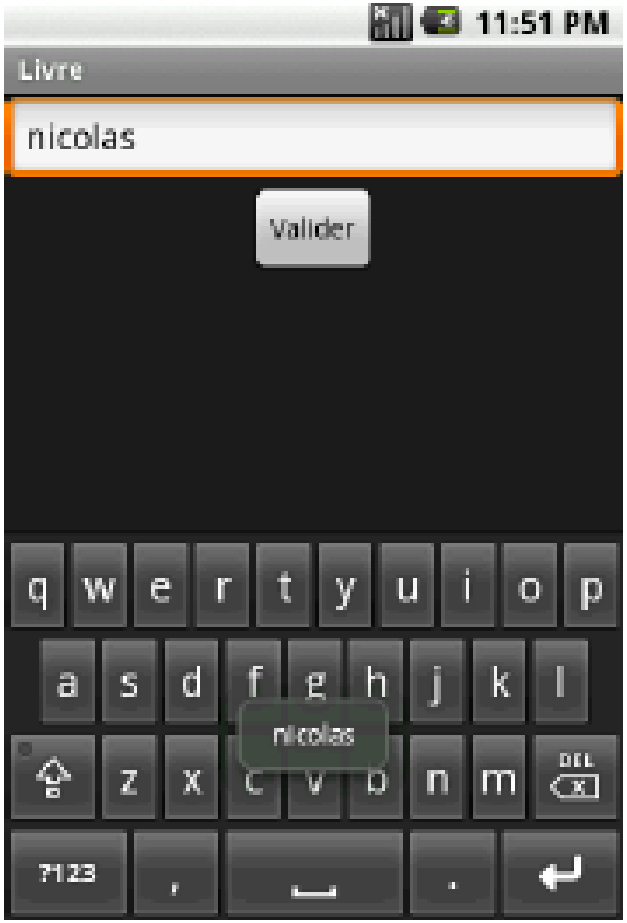
- Voyons maintenant comment récupérer ce que l'utilisateur a inscrit lorsqu'il clique sur le bouton Valider.

- Modifiez la méthode **onCreate** de l'activité Main de votre projet.

Récupérer la saisie de l'utilisateur

```
...

   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      // nous ajoutons un écouteur de type OnClickListener sur le bouton
       // de validation.
      ((Button)findViewById(R.id.monBouton)).setOnClickListener(
                                                ➥ new OnClickListener() {

         @Override
         public void onClick(View v) {

            // On récupère notre EditText
            EditText texte = ((EditText)findViewById(R.id.monEditText));
            // On garde la chaîne de caractères
            String nom = texte.getText().toString();
            // On affiche ce qui a été tapé
            Toast.makeText(Main.this, nom, Toast.LENGTH_SHORT).show();
         }
      });
   }
}

...
```

Récupérer la saisie de l'utilisateur



Intégrer d'autres composants graphiques

- Nous allons maintenant intégrer dans notre interface plusieurs types de vues que vous serez amené à utiliser dans vos applications :

- une case à cocher (**CheckBox**),
- un bouton image (**ImageButton**),
- deux boutons radio (**RadioGroup** et **RadioButton**),
- un contrôle de saisie de date (**DatePicker**),
- ……

Diverses vues dans un même gabarit

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="fill_parent"
           android:layout_height="fill_parent">
   <!-- Case a cocher -->
   <CheckBox
      android:id="@+id/CheckBox01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Conditions acceptées ?">
   </CheckBox>
   <!-- Bouton avec une Image -->
   <ImageButton
      android:id="@+id/ImageButton01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/icon">
      <!-- @drawable/icon est une Image qui se trouve dans le dossier /res/
drawable de notre projet -->
   </ImageButton>
```

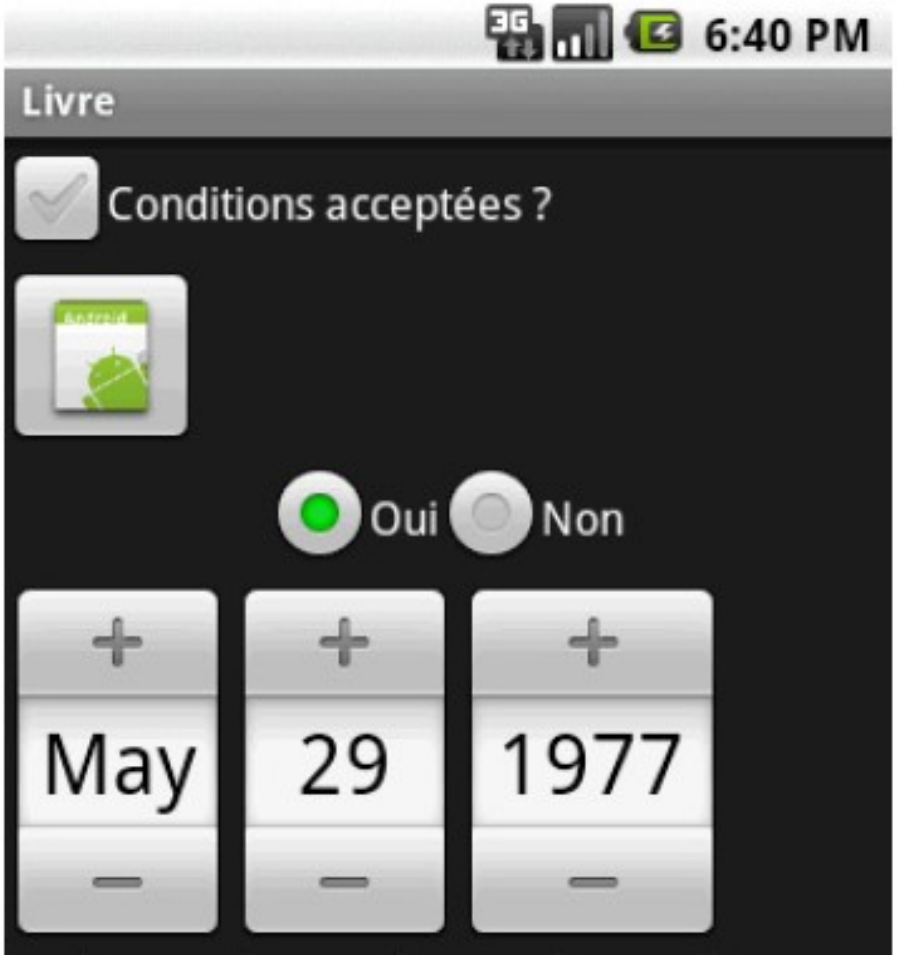
Diverses vues dans un même gabarit

```
<!-- Groupe de boutons radio -->
<RadioGroup
   android:id="@+id/RadioGroup01"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="horizontal"
   android:layout_gravity="center_horizontal">
   <!-- Radio Bouton 1 -->
   <RadioButton
      android:id="@+id/RadioButton01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Oui"
      android:checked="true">

         <!-- Nous avons mis checked=true par défaut sur notre 1er bouton
radio afin qu'il soit coché -->
      </RadioButton>
      <!-- Bouton radio 2 -->
      <RadioButton
         android:id="@+id/RadioButton02"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Non">
      </RadioButton>
   </RadioGroup>
```

Diverses vues dans un même gabarit

```
<!-- Sélectionneur de date -->
<DatePicker
    android:id="@+id/DatePicker01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</DatePicker>
```

Diverses vues dans un même gabarit



De la même façon, modifiez votre fichier `main.java` pour y placer le code suivant.

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.CheckBox;
import android.widget.DatePicker;
import android.widget.ImageButton;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;



public class Main extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      // On récupère notre case à cocher pour intercepter l'événement
      // d'état (cochée ou pas)
      ((CheckBox)findViewById(R.id.CheckBox01)).setOnCheckedChangeListener(
                  ➥ new CheckBox.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
           afficheToast("Case cochée ? : " + ((isChecked)?"Oui":"Non"));
         }
      });

      // On récupère notre sélectionneur de date (DatePicker) pour attraper
      // l'événement du changement de date
      // Attention, le numéro de mois commence a 0 dans Android, mais pas les jours.
      // Donc si vous voulez mettre le mois de Mai, vous devrez fournir 4 et non 5
      ((DatePicker)findViewById(R.id.DatePicker01)).init(1977, 4, 29,
                  ➥ new DatePicker.OnDateChangedListener() {
         @Override
         public void onDateChanged(DatePicker view, int year, int monthOfYear,
                  ➥ int dayOfMonth) {
            // On affiche la nouvelle date qui vient d'être changée dans notre
            // DatePicker
            afficheToast("La date a changé\nAnnée : " + year + " | Mois : "
                  ➥ + monthOfYear + " | Jour : " + dayOfMonth);
         }
      });
```

```
        // On récupère notre groupe de bouton radio pour attraper le choix de
        // l'utilisateur
        ((RadioGroup)findViewById(R.id.RadioGroup01)).setOnCheckedChangeListener(
                        ➥ new RadioGroup.OnCheckedChangeListener() {
           @Override
           public void onCheckedChanged(RadioGroup group, int checkedId) {
              // On affiche le choix de l'utilisateur
              afficheToast("Vous avez répondu : "
                        ➥ + ((RadioButton)findViewById(checkedId)).getText());
           }
        });

        // On récupère notre Bouton Image pour attraper le clic effectué par
        // l'utilisateur
        ((ImageButton)findViewById(R.id.ImageButton01)).setOnClickListener(
                        ➥ new OnClickListener() {
           @Override
           public void onClick(View v) {
              // On affiche un message pour signalé que le bouton image a été pressé
              afficheToast("Bouton Image pressé");
           }
        });
     }

     // Méthode d'aide qui simplifiera la tâche d'affichage d'un message
     public void afficheToast(String text)
     {
        Toast.makeText(this, text, Toast.LENGTH_SHORT).show();
     }
}
```

Dans cet exemple, nous employons diverses vues que vous serez souvent amené à utiliser :

- la case à cocher (`CheckBox`) : propose un choix basique à vos utilisateurs : « oui » ou « non ». Pour récupérer à tout moment l'état de la case à cocher, il vous suffit de le récupérer avec la méthode `isChecked`. Dans notre exemple nous récupérons l'état dès qu'il change, grâce à l'écouteur `OnCheckedChangeListener` que nous

avons associé à la case à cocher. Ceci nous permet de récupérer immédiatement le changement d'état et si nécessaire d'exécuter du code en conséquence ;

- le bouton image (`ImageButton`) : fonctionne de la même façon qu'un bouton classique mais présenté précédemment sous la forme de l'image choisie. La principale différence avec un bouton classique est que l'on ne peut ajouter un texte à afficher : seule l'image est affichée ;
- le groupe de boutons radio et les boutons radio (`RadioGroup` et `RadioButton`) : propose une liste de choix à réponse unique (seul un des boutons du groupe de boutons pourra être coché à un instant *t*). Afin que les boutons radio (de type `RadioButton`) puissent basculer de l'un à l'autre, vous devez les regrouper dans un élément `ViewGroup` de type `RadioGroup`. Si vos boutons radio ne sont pas regroupés dans un `RadioGroup,` ils seront tous indépendants. Dans notre exemple, nous attrapons le changement d'état des boutons radio dès qu'ils sont pressés grâce à l'écouteur `OnCheckedChangeListener`que nous avons associé au `RadioGroup`. Si vous souhaitez récupérer le bouton radio pressé à n'importe quel moment il vous suffira d'appeler la méthode `getCheckedRadioButtonId` ;
- le sélectionneur de date (`DatePicker`) : affiche les boutons nécessaires pour saisir une date. Attention celui-ci s'adapte visuellement à la langue et région du téléphone. Ainsi, un téléphone en langue « Anglais US » affiche la date sous la forme Mois-Jour-Année tandis que le même téléphone en langue française l'affiche sous la forme Jour-Mois-Année. Dans notre exemple, nous récupérons le changement de date de notre sélectionneur de date dès que l'utilisateur modifie une valeur grâce à l'écouteur `OnDateChangedListner` que nous lui avons associé. Pour récupérer les valeurs sélectionnées, vous pouvez à tout moment utiliser les méthodes`getDayOfMonth`, `getMonth`, `getYear`. Attention, comme signalé en commentaire dans l'exemple, le numéro des mois commence à 0 (janvier = 0, février = 1 etc.) ce qui n'est pas le cas pour les jours et les années ;