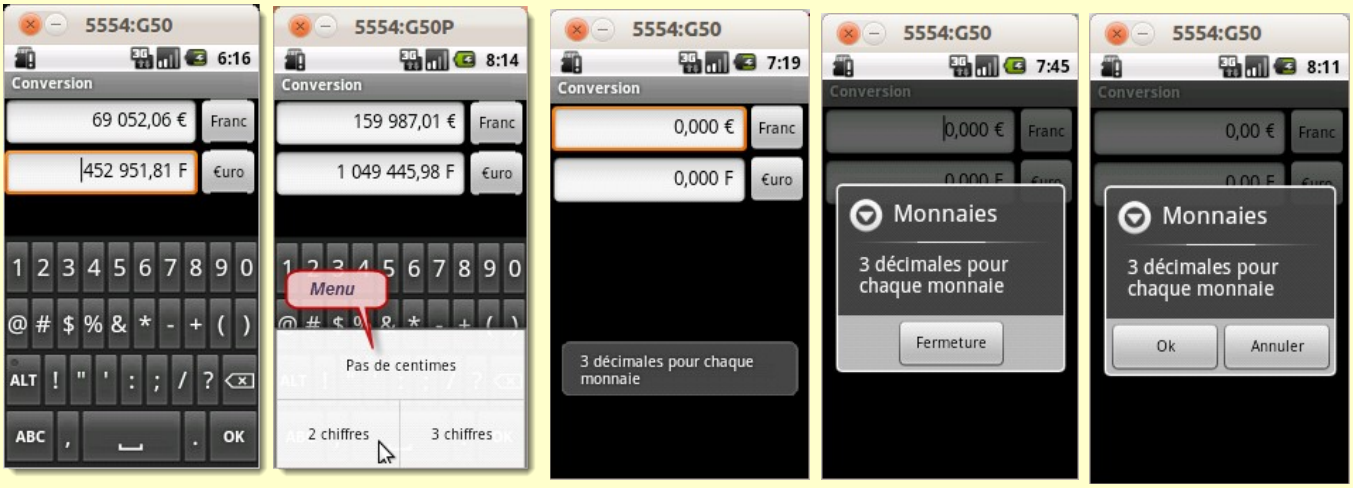
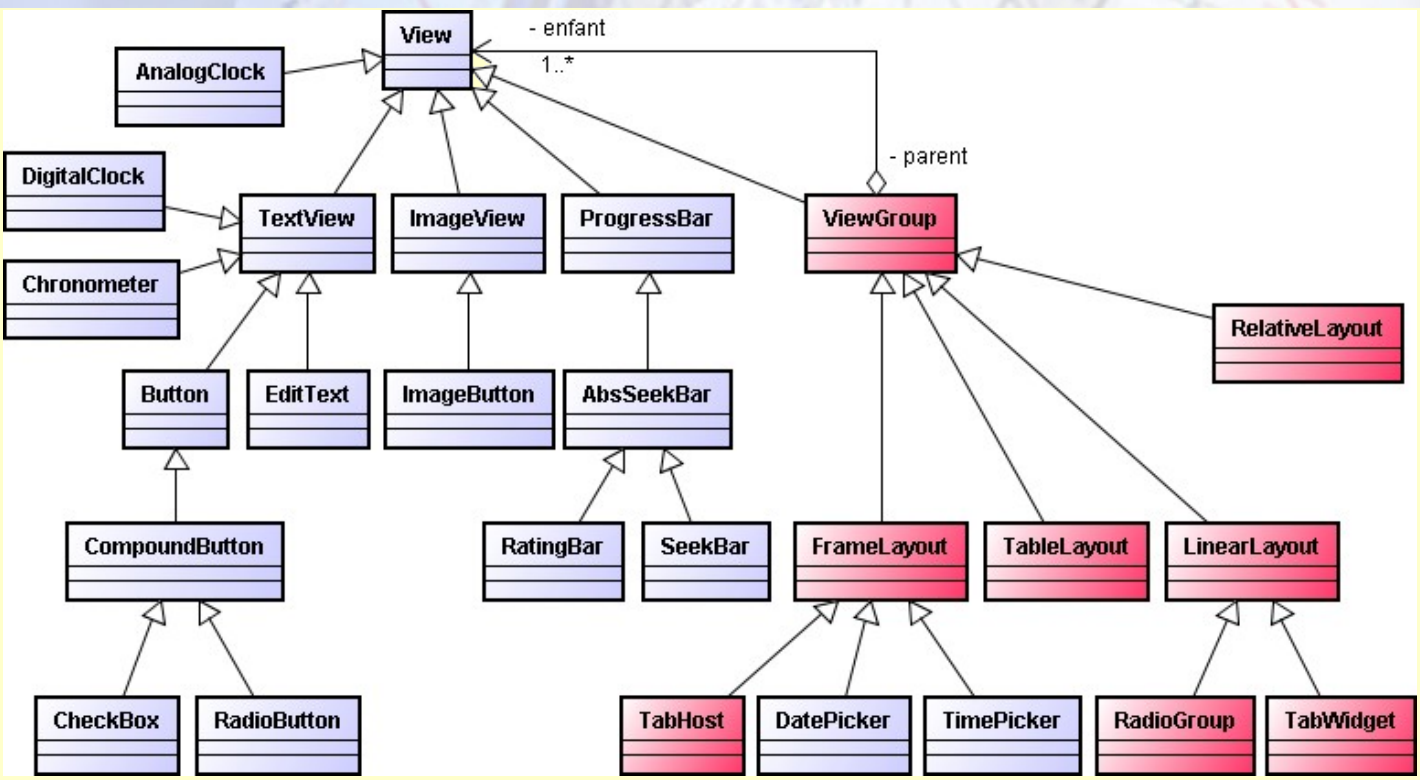
Cette étude va nous permettre de conforter nos connaissances d'Android en enrichissant votre activité principale d'un certain nombre d'éléments annexes qui peuvent s'avérer bien sympathiques :

- *Création d'un nouveau composant.*
- *Création d'un menu pour proposer différents réglages sur l'activité principale.*
- *Mise en place de messages surgissant.*



CRÉATION D'UN NOUVEAU COMPOSANT

Nous allons reprendre le projet de conversions monétaires et nous allons créer un nouveau composant qui est capable de saisir et d'afficher directement des valeurs monétaires. Nous décidons donc d'étendre la vue ***EditText*** afin de permettre une saisie, et nous appelons ce nouveau composant ***Monnaie.***



Nota : *Pour créer un composant personnalisé, la première étape est de déterminer quelle est la classe la plus adaptée pour former la base de votre composant.*

fr.manu.monnaie.Monnaie.java


```
package fr.btsiris.monnaie;
import android.content.Context;
import android.text.InputType;
import android.util.AttributeSet;
import android.view.Gravity;
import android.widget.EditText;
import java.text.*;
public class Monnaie extends EditText {
    private NumberFormat formatMonnaie;
    public Monnaie(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init();
    }

    public Monnaie(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public Monnaie(Context context) {
        super(context);
        init();
    }

    private void init() {
        setSymbole('€');
        setValeur(0.0);
        setGravity(Gravity.RIGHT);
        setInputType(InputType.TYPE_CLASS_NUMBER);
    }

    public void setSymbole(char symbole) { formatMonnaie = new DecimalFormat("#,##0.00 "+symbole);  }

    public double getValeur() {
        try {
            Number valeur = (Number) formatMonnaie.parse(getText().toString());
            setText(formatMonnaie.format(valeur));
            return valeur.doubleValue();
        }
        catch (ParseException ex) { return 0.0; }
    }

    public void setValeur(double valeur) { setText(formatMonnaie.format(valeur));  }
}
```

- Vous noterez la déclaration **des trois constructeurs de base d'une vue**. Ils servent notamment à créer une vue à partir d'un gabarit d'interface au format XML. De cette façon, la création du composant et son utilisation dans un fichier de description XML sont semblables à tout autre composant de la plate-forme Android.
- Vous remarquez également la présence de méthodes supplémentaires accesseurs (**setXxx()** notamment) permettant de mettre à jour les propriétés du composant.

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1">
    <TableRow>
        <fr.btsiris.monnaie.Monnaie
            android:id="@+id/euro"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_span="3"/>
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Franc"
            android:onClick="calculFranc"  />
    </TableRow>

    <TableRow>
        <fr.btsiris.monnaie.Monnaie
```

```
            android:id="@+id/franc"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_span="3"/>
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="€uro"
            android:onClick="calculEuro"  />
    </TableRow>
</TableLayout>
```

Nota : *Pour faire référence à notre nouveau composant dans le fichier de description XML concernant le layout, il suffit d'utiliser la balise qui porte son nom. La seule petite contrainte est de préciser le nom complet avec son chemin, ici donc* ***<fr.btsiris.monnaie.Monnaie>.***

fr.manu.monnaie.Conversion.java

```
package fr.btsiris.monnaie;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
public class Conversion extends Activity {
    private Monnaie euro, franc;
    private final double TAUX = 6.55957;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        euro = (Monnaie) findViewById(R.id.euro);
        franc = (Monnaie) findViewById(R.id.franc);
        franc.setSymbole('F');
        franc.setValeur(0.0);
    }

    public void calculFranc(View vue) { franc.setValeur(euro.getValeur() * TAUX);    }
    public void calculEuro(View vue) {  euro.setValeur(franc.getValeur() / TAUX);   }
}
```

CRÉER DES MENUS POUR VOS ACTIVITÉS

Tous les modèles d'Android possèdent un bouton Menu. Grâce à ce bouton, il vous est possible de proposer à vos utilisateurs des fonctionnalités supplémentaires n'apparaissant pas par défaut à l'écran afin de mieux gérer la taille limitée de l'écran d'un appareil mobile.

Nota : *Chaque menu est propre à une activité, c'est pourquoi toutes les opérations que nous allons traiter dans cette partie se réfère à une activité. Pour proposer plusieurs menus vous aurez donc besoin de le faire dans chaque activité de votre application.*

Un menu se déclenche en appuyant sur le bouton Menu du terminal et possède deux modes de fonctionnement : **icône** et **étendu**.

- *Lorsque l'utilisateur appuie sur le bouton Menu, le menu est en* ***mode icône*** *et n'affiche que les six premiers choix sous la forme de gros boutons faciles à sélectionner, disposés en ligne en bas de l'écran.*
- *Si ce menu compte plus de six choix, le sixième bouton affiche* ***"Plus"*** *- cliquez sur cette option fait passer le menu en* ***mode étendu****, qui affiche tous les choix restants. L'utilisateur peut bien sûr faire défiler le menu afin d'effectuer n'importe quel choix.*

Pour créer un menu, *il vous suffit de redéfinir la méthode de rappel, prévue à cette effet,* ***onCreateOptionsMenu()*** *de la classe* ***Activity****. Cette méthode est effectivement appelée la première fois que l'utilisateur appuie sur le bouton Menu de son téléphone. Elle reçoit en paramètre un objet de type* ***Menu*** *dans lequel nous ajoutons nos différents choix.*

Ajout des différents options dans votre menu *: Pour ajouter toutes les sélections possibles à votre menu, utilisez la méthode* ***add()*** *de la classe* ***Menu****. Celle-ci est surchargée afin de recevoir les combinaisons des paramètres suivants :*

- ***Un identifiant de groupe*** *(un entier) qui doit valoir* ***NONE*** *(ou tout simplement la valeur entière* ***0****), sauf si vous désirez créer un ensemble d'options de menu qui regroupe un même type de sélection, au moyen de la méthode* ***setGroupCheckable()****.*
- ***Un identifiant de choix*** *(également un entier) servant à identifier la sélection dans la méthode de rappel* ***onOptionsItemSelected()*** *lorsqu'une option du menu a été choisie.*
- ***Un identifiant d'ordre*** *(encore un entier), indiquant l'emplacement du choix dans le menu lorsque ce dernier contient des options ajoutées par Android. Sans cela, contentez-vous d'utiliser la constante* ***NONE****.*
- ***Le texte du choix****, sous la forme d'une chaîne de caractères de type* ***String*** *ou d'un identifiant de ressource.*

PROPOSER UN MENU À NOTRE CONVERSION MONÉTAIRE

Nous reprenons l'exemple qui réalise des conversions monétaires dans les deux sens. Cette fois-ci, il sera possible de choisir le nombre de chiffres pour les centimes, soit deux, soit trois (des millièmes) ou tout simplement sans montrer les centimes. *Ici, rien n'est changé au niveau du fichier de description XML.*

fr.manu.monnaie.Monnaie.java

```
package fr.btsiris.monnaie;
import android.content.Context;
import android.text.InputType;
import android.util.AttributeSet;
import android.view.Gravity;
import android.widget.EditText;
import java.text.*;
public class Monnaie extends EditText {
    private NumberFormat formatMonnaie;
    private String nombreDecimales = "0.00 ";
    private char symbole;

…

    public void setSymbole(char symbole) {
        this.symbole = symbole;
        formatMonnaie = new DecimalFormat("#,##"+nombreDecimales+symbole);
    }

    public void setNombreDecimales(int nombre) {
        switch(nombre) {
            case 0 : nombreDecimales = "0 "; break;
            case 2 : nombreDecimales = "0.00 "; break;
            case 3 : nombreDecimales = "0.000 "; break;
        }
        formatMonnaie = new DecimalFormat("#,##"+nombreDecimales+symbole);
        getValeur();
    }
}
```

- *La grande nouveauté par rapport au code précédent, c'est la présence d'une nouvelle propriété* ***nombreDécimales*** *qui va servir au formatage du texte suivant la valeur monétaire désirée et bien sûr suivant le nombre de décimales après la virgule.*
- *Pour que tout se passe correctement, il est aussi nécessaire cette fois-ci d'enregistrer le symbole de la monnaie.*

> ***Pour l'activité principale,*** *nous devons maintenant mettre en œuvre notre menu au moyen des deux méthodes importantes, savoir* ***onCreateOptionsMenu()*** *et* ***onOptionsItemSelected()****.*

fr.manu.monnaie.Conversion.java

```
package fr.btsiris.monnaie;

public class Conversion extends Activity {
    private Monnaie euro, franc;
    private final double TAUX = 6.55957;

    …

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(Menu.NONE, 0, Menu.NONE, "Pas de centimes");
        menu.add(Menu.NONE, 2, Menu.NONE, "2 chiffres");
        menu.add(Menu.NONE, 3, Menu.NONE, "3 chiffres");
        return super.onCreateOptionsMenu(menu);   // ou bien return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        euro.setNombreDecimales(item.getItemId());
        franc.setNombreDecimales(item.getItemId());
        return super.onOptionsItemSelected(item);   // ou bien return true;
    }
}
```

✗ AFFICHAGE DE MESSAGES SURGISSANT

Dans certaines situations, vous pouvez avoir besoin de communiquer avec l'utilisateur par des messages d'alerte ou de confirmation en vous affranchissant des limites de l'interface classique. Android dispose de plusieurs moyens d'alerter les utilisateurs par d'autres systèmes que ceux des activités classiques. Deux principales méthodes permettent de faire surgir des messages, les **toasts** et les **alertes**.

Les toasts *: Un* ***toast*** *est un message transitoire, ce qui signifie qu'il s'affiche et disparaît de lui-même, sans intervention de l'utilisateur. En outre, il ne modifie pas le focus de l'activité courante ; si l'utilisateur est en train de réaliser une saisie, ses frappes au clavier ne seront pas capturées par le message surgissant.*

✗ *Un* ***toast*** *étant transitoire, vous n'avez aucun moyen de savoir si l'utilisateur l'a remarqué. Vous ne recevrez aucun accusé de réception de sa part et le message ne restera pas affiché suffisamment longtemps pour ennuyer l'utilisateur. Un* ***toast*** *est donc essentiellement conçu pour diffuser des messages d'avertissement - pour annoncer qu'une longue tâche en arrière-plan s'est terminée, que la batterie est presque vide, etc.*

✗ *La création d'un* ***toast*** *est assez simple. Il existe pour cela la classe* ***Toast*** *qui fournit une méthode statique* ***makeText()*** *qui prend un objet* ***String*** *(ou un identifiant d'une ressource textuelle) en paramètre et qui renvoie une instance de* ***Toast****. Les autres paramètres de cette méthode sont l'activité (ou tout autre contexte) et une durée valant* ***LENGTH_SHORT*** *ou* ***LENGTH_LONG*** *pour exprimer la durée relative pendant laquelle le message restera visible. Lorsque le toast est configuré, il suffit d'appeler sa méthode* ***show()*** *pour qu'il s'affiche.*

fr.manu.monnaie.Conversion.java

```
package fr.btsiris.monnaie;
import android.app.Activity;
import android.os.Bundle;
import android.view.*;
public class Conversion extends Activity {
    private Monnaie euro, franc;
    private final double TAUX = 6.55957;

    . . .

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(Menu.NONE, 0, Menu.NONE, "Pas de centimes");
        menu.add(Menu.NONE, 2, Menu.NONE, "2 chiffres");
        menu.add(Menu.NONE, 3, Menu.NONE, "3 chiffres");
        return super.onCreateOptionsMenu(menu);   // ou bien return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        euro.setNombreDecimales(item.getItemId());
        franc.setNombreDecimales(item.getItemId());
        String decimales = item.getItemId() == 0 ? "Pas de centimes" : ""+ item.getItemId() + " décimales pour chaque monnaie";
        Toast.makeText(this, decimales, Toast.LENGTH_SHORT).show();
        return super.onOptionsItemSelected(item);   // ou bien return true;
    }
}
```

Les alertes *:Si vous préférez utiliser le style plus classique des boîtes de dialogue, choisissez plutôt* ***AlertDialog****. Comme toute boîte de dialogue modale, un* ***AlertDialog*** *s'ouvre, prend le focus et reste affiché tant que l'utilisateur ne le ferme pas.*

✗ *Ce type d'affichage convient donc bien aux erreurs critiques, aux messages de validation qui ne peuvent pas être affichés correctement dans l'interface de base de l'activité ou à toute autre information dont vous voulez vous assurer la lecture immédiate par l'utilisateur.*

Pour créer un ***AlertDialog****, le moyen le plus simple consiste à utiliser la classe* ***Builder*** *qui offre un ensemble de méthodes permettant de configurer un* ***AlertDialog****. Chacune de ces méthodes renvoie une instance de* ***Builder*** *afin de faciliter le chaînage des appels. A la fin, il suffit d'appeler la méthode* ***show()*** *de l'objet* ***Builder*** *pour afficher la boîte de dialogue. Après l'appel de* ***show()****, la boîte de dialogue s'affiche et attend une intervention de l'utilisateur. Voici les méthodes de configuration de* ***Builder*** *les plus utilisées :*

✗ ***setMessage()*** *: permet de définir le corps de la boîte de dialogue à partir d'un simple message de texte. Son paramètre est un objet* ***String*** *ou un identifiant de ressource textuelle.*

✗ ***setTitle()*** *et* ***setIcon()*** *: permettent de configurer le texte et/ou l'icône qui apparaîtra dans la barre de titre de la boîte de dialogue.*

✗ ***setPositiveButton(), setNeutralButton()*** *et* ***setNegativeButton()*** *: permettent d'indiquer les boutons qui apparaîtrons en bas de la boîte de dialogue, leur emplacement latéral (respectivement, à gauche, au centre ou à droite), leur texte et le code qui sera appelé lorsque nous cliquons sur un bouton (en plus de refermer la boîte de dialogue).*

fr.manu.monnaie.Conversion.java

```
package fr.btsiris.monnaie;
. . .
public class Conversion extends Activity {
    private Monnaie euro, franc;
    private final double TAUX = 6.55957;

. . .

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(Menu.NONE, 0, Menu.NONE, "Pas de centimes");
        menu.add(Menu.NONE, 2, Menu.NONE, "2 chiffres");
        menu.add(Menu.NONE, 3, Menu.NONE, "3 chiffres");
        return super.onCreateOptionsMenu(menu);   // ou bien return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        euro.setNombreDecimales(item.getItemId());
        franc.setNombreDecimales(item.getItemId());
        String decimales = item.getItemId() == 0 ? "Pas de centimes" : ""+ item.getItemId() + " décimales pour chaque monnaie";
        new AlertDialog.Builder(this).setTitle("Monnaies").setMessage(decimales).setNeutralButton("Fermeture", null).show();
        return super.onOptionsItemSelected(item);   // ou bien return true;
    }
}
```

fr.manu.monnaie.Conversion.java

```
package fr.btsiris.monnaie;
. . .
public class Conversion extends Activity {
    private Monnaie euro, franc;
    private final double TAUX = 6.55957;
. . .

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
     final MenuItem choix = item;
     String decimales = item.getItemId() == 0 ? "Pas de centimes" : ""
                        + item.getItemId() + " décimales pour chaque monnaie";
     new AlertDialog.Builder(this).setTitle("Monnaies").setMessage(decimales)
             .setPositiveButton("Ok", new DialogInterface.OnClickListener() {
                 public void onClick(DialogInterface di, int i) {
                     euro.setNombreDecimales(choix.getItemId()) ;
                     franc.setNombreDecimales(choix.getItemId());
                 }
             }).setNegativeButton("Annuler", null).show();
     return super.onOptionsItemSelected(item);   // ou bien return true;
  }
}
```