UVS
UNIVERSITE VIRTUELLE DU SENEGAL

Notions essentielles d'une application android

Initiation au développement mobile

Ndiogou THIAO

plan

Sequence 3 : Nations essentielles d'une application Android

La plate-forme android

Rappelons les points clés d'Android en tant que plate-forme :

- elle est innovante car toutes les dernières technologies de téléphonie y sont intégrées : écran tactile, accéléromètre, GPS, appareil photo numérique etc.

- elle est accessible car en tant que développeur vous n'avez pas à acheter de matériel spécifique (si vous voulez aller plus loin que l'utilisation d'un émulateur, un téléphone Android pour effectuer vos tests vous sera toutefois nécessaire), ni à connaître un langage peu utilisé ou spécifique : le développement sur la plateforme Android est en effet réalisé en langage Java, un des langages de programmationles plus répandus ;

- elle est ouverte parce la plate-forme Android est fournie sous licence open source, permettant à tous les développeurs – et constructeurs – de consulter les sources et d'effectuer les modifications qu'ils souhaitent.

Les versions de androide (1/2)

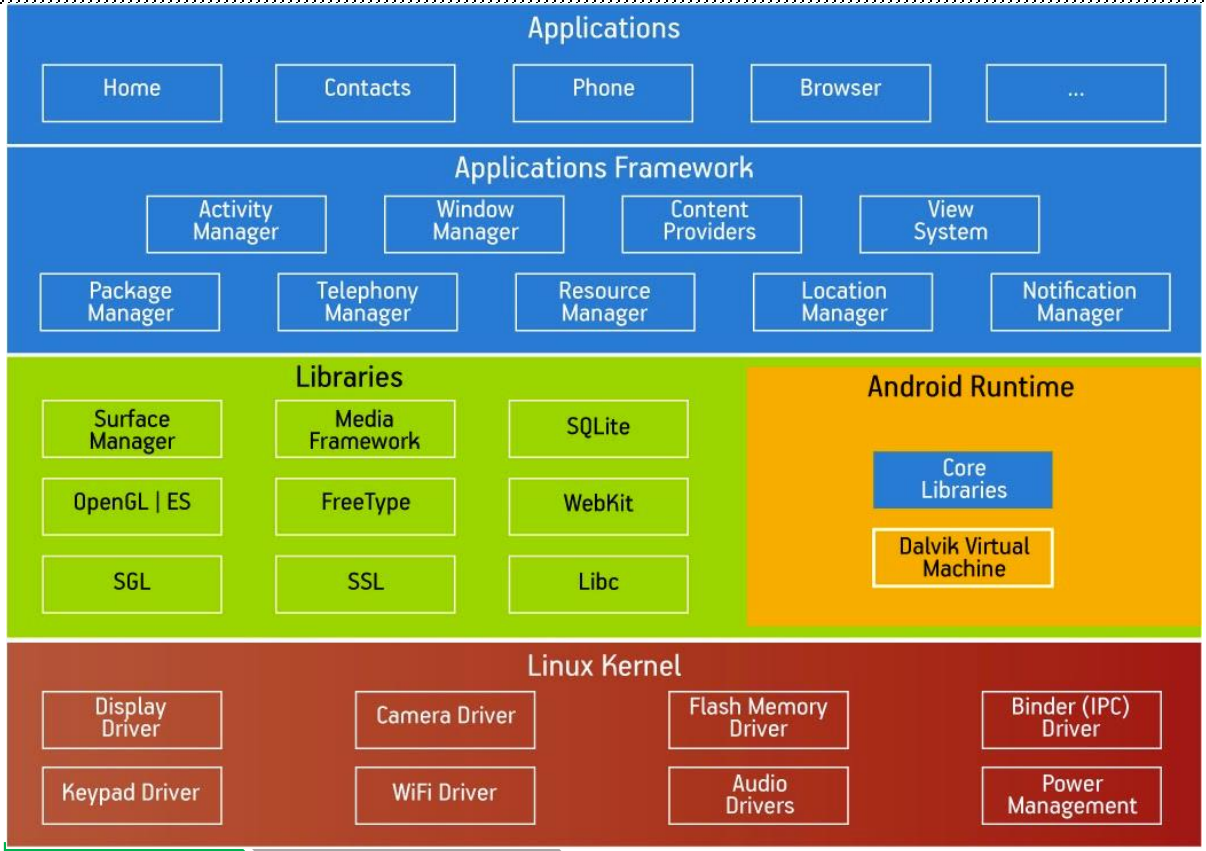
- Les versions android se succèdent rapidement et les changements qui les accompagnent sont souvent conséquents en termes de nouvelles fonctionnalités et d'améliorations. Cependant, ces évolutions n'ont pas toujours été sans impact sur le bon fonctionnement et la compatibilité des applications entre versions.

Les versions de androide (2/2)

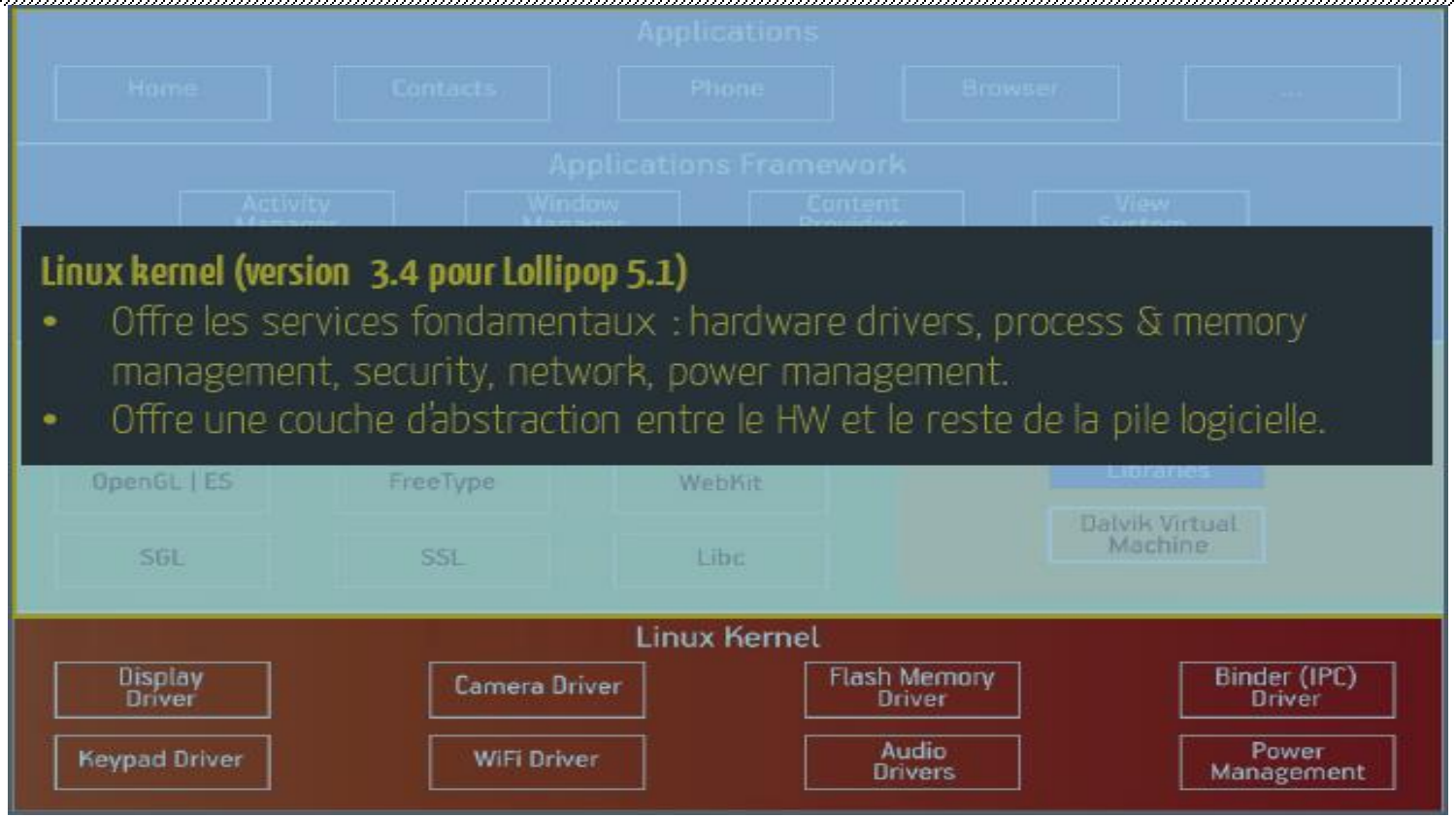
Tableau des versions

CodeName	Platform	API Level
Cupcake	Android 1.5	3
Donut	Android 1.6	4
Eclair	Android 2.1	7
Froyo	Android 2.2	8
Gingerbread	Android 2.3	9
Honeycomb	Android 3.0	11
Ice Cream Sandwich	Android 4.0	14
Jelly Bean	Android 4.1	16
KitKat	Android 4.4	19
Lollipop	Android 5.0	20
Marshmallow	Android 6.0	23
Nougat	Android 7.0	24

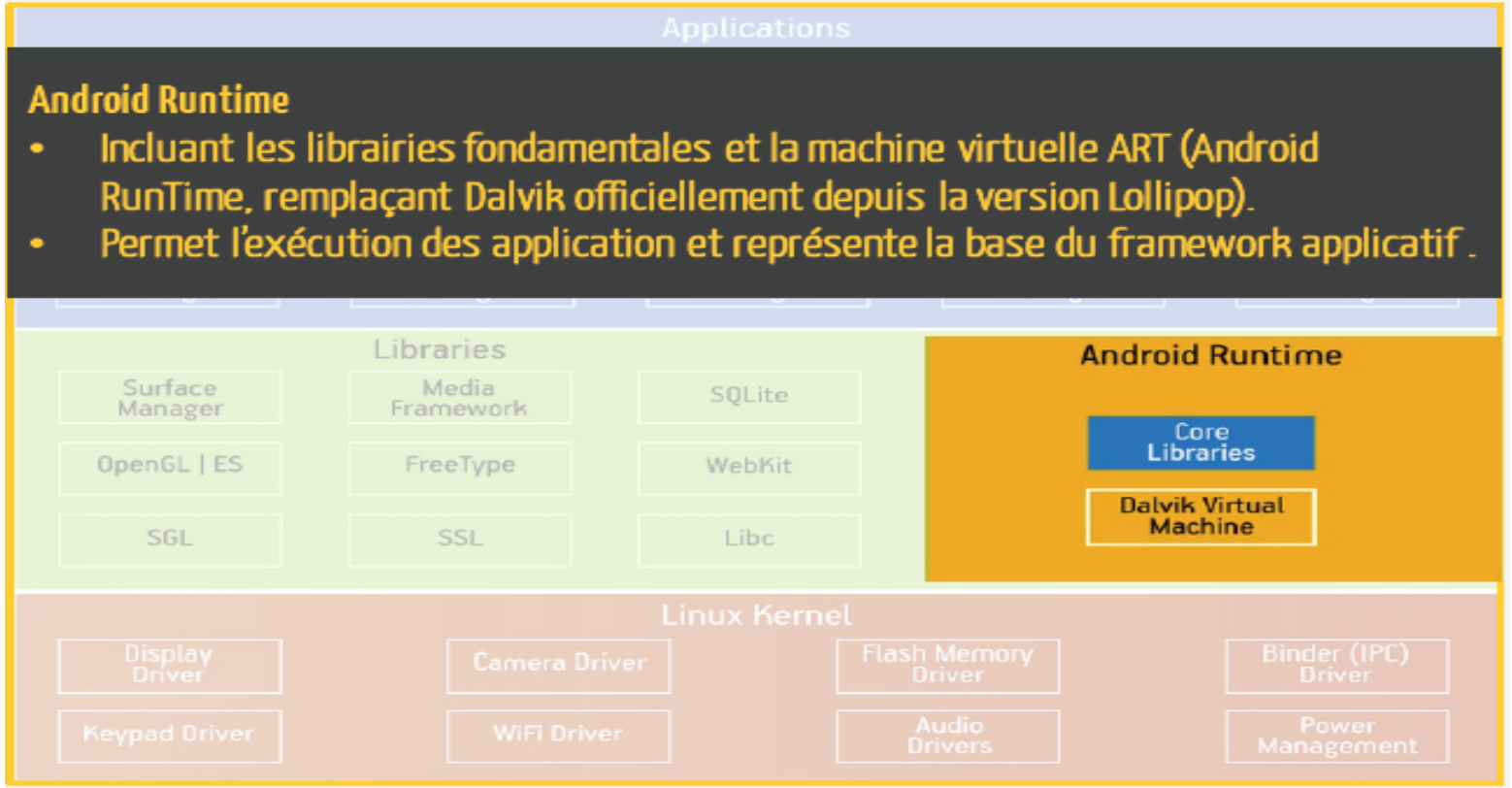
Architecture Android (1/5)



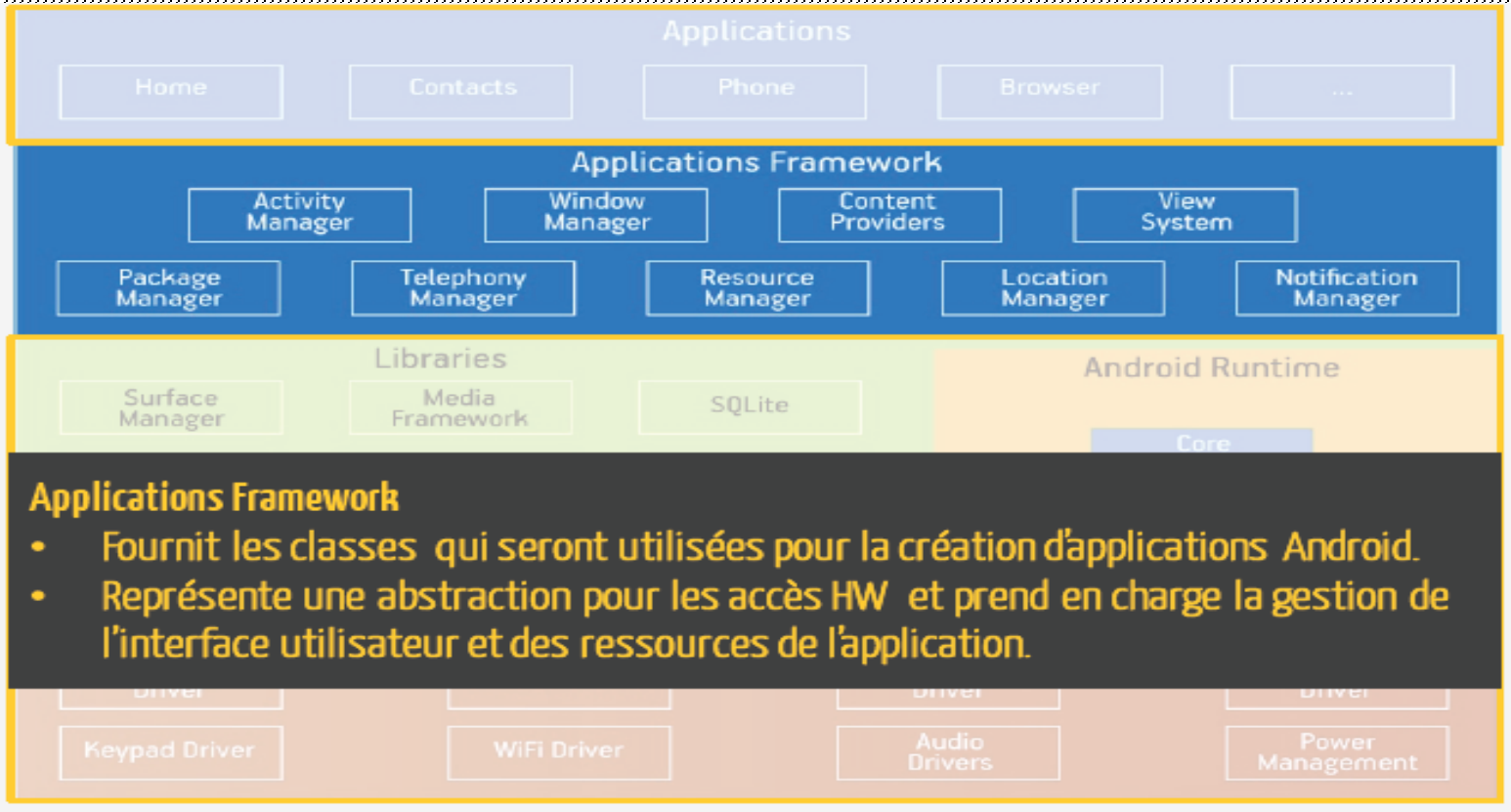
Architecture Android (2/5)



Architecture Android(3/5)



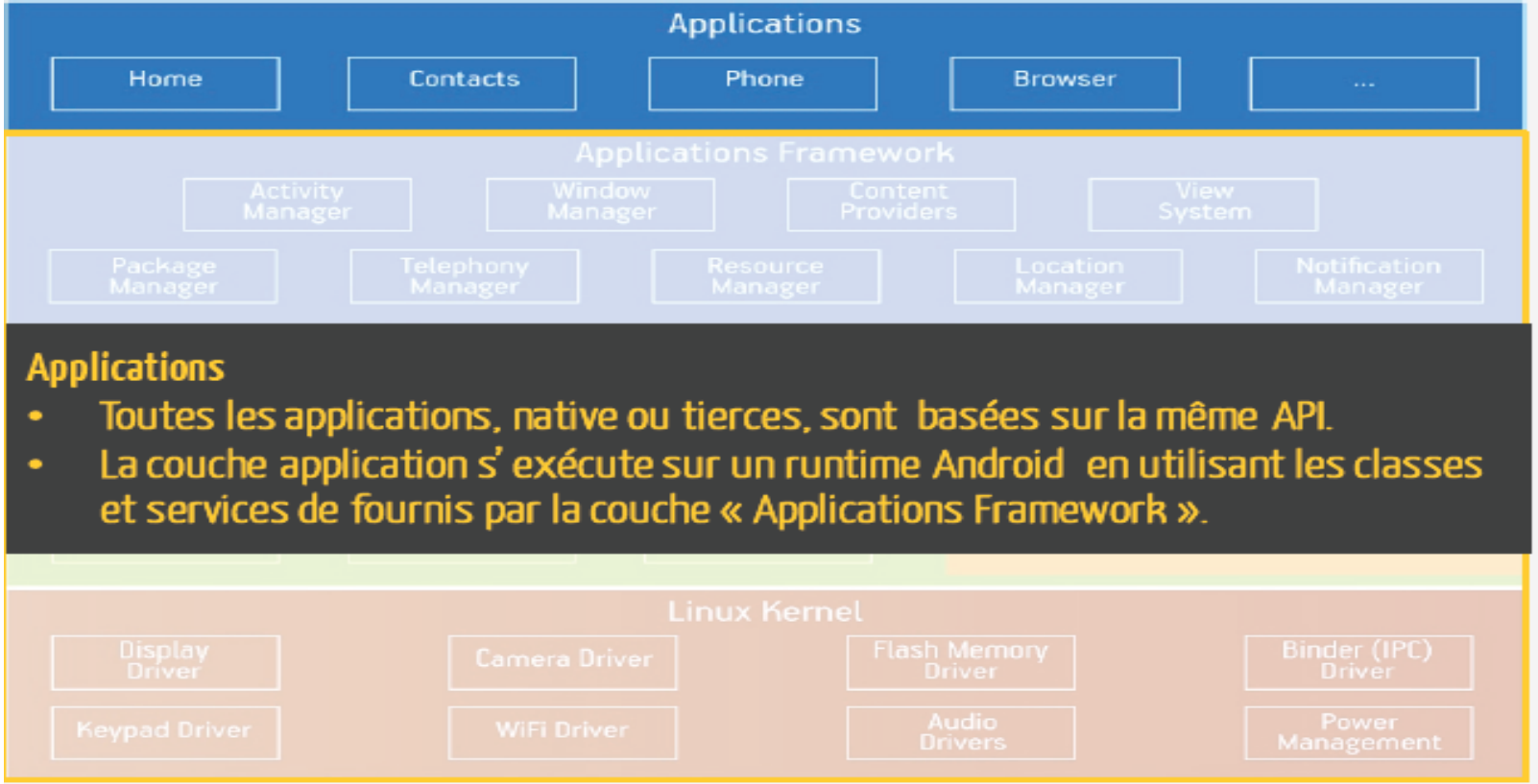
Architecture Android(4/5)



Applications Framework

- Fournit les classes qui seront utilisées pour la création d'applications Android.
- Représente une abstraction pour les accès HW et prend en charge la gestion de l'interface utilisateur et des ressources de l'application.

Architecture Android (5/5)



Applications

- Toutes les applications, native ou tierces, sont basées sur la même API.
- La couche application s' exécute sur un runtime Android en utilisant les classes et services de fournis par la couche « Applications Framework ».

Les composant d'une application Androide(1/8)

Les composants applicatifs : (activités)

- L'*activité* représente le bloc de base d'une application. Elle correspond à la partie présentation

l'application et fonctionne par le biais de vues qui affichent des interfaces graphiques et répondent aux actions utilisateur

- Toute application, à l'exception des applications de type services, a au moins une activité dont la première est celle qu'on voit au démarrage de l'application.
- Déclaration et Implantation d'une activité
- L'implantation d'une activité se fait en :
  - créant une classe qui étend android.app.Activity
  - déclarant l'activité dans le fichier de manifeste
  - création du fichier de layout associé à l'activité
- Fort heureusement, Android Studio crée le squelette de l'activité, une entrée de l'Activité dans le manifeste est le fichier de vue dont le nom est bâti à partir du non de l'activité -d'un seul coup.

Les composant d'une application Androide (2/8)

Les composants applicatifs : (service)

- Le *service* est un composant qui fonctionne en tâche de fond, de manière invisible. Ses principales utilisations sont la mise à jour de sources de données ainsi que d'activités visibles et le déclenchement de notifications. Le chapitre traitant des services et de la gestion des threads présente en détail leur utilisation.

- La déclarations se fait comme ainsi:

Les composant d'une application Androide (3/8)

Les composants applicatifs :(Service)

SYNTAX:

```
<service android:description="string resource"
         android:directBootAware=["true" | "false"]
         android:enabled=["true" | "false"]
         android:exported=["true" | "false"]
         android:icon="drawable resource"
         android:isolatedProcess=["true" | "false"]
         android:label="string resource"
         android:name="string"
         android:permission="string"
         android:process="string" >
    . . .
</service>
```

CONTENU DANS :

<application>

PEUT CONTENIR:

<intent-filter>

<meta-data>

Déclare un service (une sous-classe de service) comme l'un des composants de l'application. Tous les services doivent être représentés par des éléments <service> dans le fichier manifeste. Tout ce qui n'est pas déclaré ne sera pas vu par le système et ne sera jamais exécuté.

Les composant d'une application Android e (4/8)

Les composants applicatifs : (les fournisseurs de contenu)

- Le *fournisseur de contenu* permet de gérer et de partager des informations.
- Un même fournisseur permet d'accéder à des données au sein d'une application et entre applications.

- Le *gadget* est un composant graphique qui s'installe sur le bureau Android. Le calendrier qui affiche de l'information ou le lecteur audio qui permet de contrôler la lecture de fichiers sont deux exemples de gadgets que l'on trouve souvent sur un écran d'accueil.

Les composant d'une application Android e (5/8)

Les différents composants applicatifs Android et les classes associées

Non	Classe ou paquetage concerné(e)
Activité	android.app.Activity
Services	android.app.Service
Fournisseur de contenu	android.content.ContentProvider
Gadget	android.appwidget.*

Les composant d'une application Androide (6/8)

Éléments d'interaction : (intents)

- *L'objet Intent* : il permet de diffuser des messages en demandant la réalisation d'une action .
- L'accès aux autres applications et au système étant restreinte par le modèle de sécurité Android, ces objets permettent aux applications de fournir ou demander des services ou des données.
- La transmission se fait à travers tout le système et peut cibler précisément une activité ou un service.

.

Les composant d'une application Androide (7/8)

Éléments d'interaction : récepteurs, notifications

- *Récepteur d'Intents* : il permet à une application d'être à l'écoute des autres afin de répondre aux objets Intent qui lui sont destinés et qui sont envoyés par d'autres composants applicatifs.

- *Notification* : une notification signale une information à l'utilisateur sans interrompre ses actions en cours.

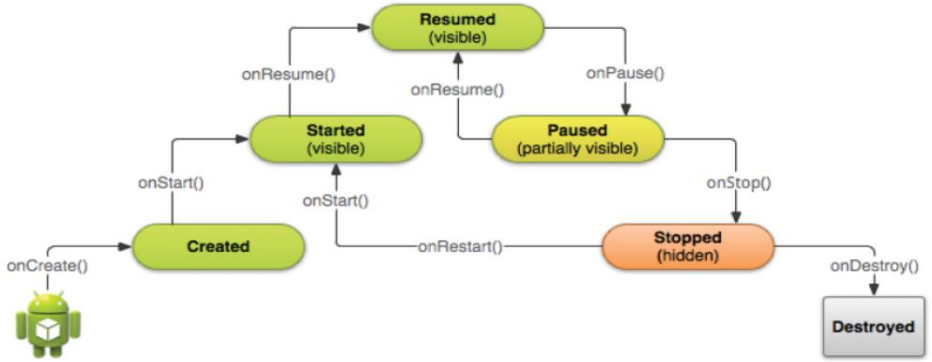
Les composant d'une application Androide (8/8)

Éléments d'interaction : intents, récepteurs, notifications

Non	Classe concernée
Intent	android.content.Intent
Récepteur d'Intents *(Broadcast Receiver)*	android.content.BroadcastReceiver
Notification (*Notification*)	android.app.Notification

Cycle de vie dune activité (1/3)

- Aucune méthode main dans un programme Android
- Android exécute le code d'une activité en appelant des *callbacks*
- Ces callbacks correspondent aux phrases de la vie d'une activité
- Il n'est pas nécessaire d'implémenter toutes les callbacks



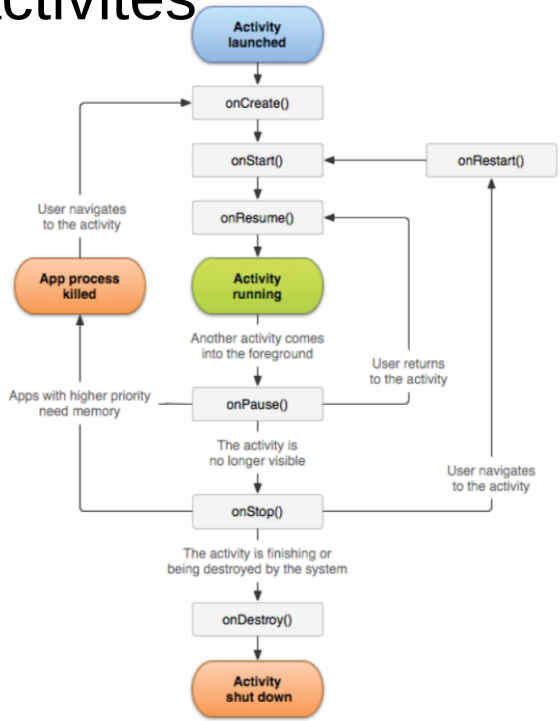
Cycle de vie dune activité (2/3)

États durables d'une activité

Méthode	signification
onCreate()	Invoqué quand Called when the activity is first created
onStart()	Appelé juste après sa création ou par la méthode de redémarrage après onStop (). Ici l'activité commence devenir visible pour l'utilisateur
onResume()	Appelé lorsque l'activité est visible pour l'utilisateur et que l'utilisateur peut interagir avec celui-ci
onPause()	Appelé lorsque le contenu de l'activité n'est pas visible car l'utilisateur reprend l'activité précédente
onStop()	Appelé lorsque l'activité n'est pas visible pour l'utilisateur car une autre activité l'a remplacé
onRestart()	Appelé lorsque l'utilisateur arrive à l'écran ou reprend l'activité qui a été arrêtée
onDestroy()	Appelé lorsque l'activité n'est pas en arrière-plan

Cycle de vie dune activité (3/3)

Cycle de vie des activités



Les interfaces utilisateur

- Les interfaces graphiques prennent une place de plus en plus importante dans le des applications par les utilisateurs, tant dans l'implémentation de concepts innovants qu'au niveau de l'ergonomie.

- Comme sur bien des plates-formes, les interfaces d'applications Android sont organisées en vues et gabarits, avec néanmoins quelques spécificités

Les interfaces utilisateur

Le concépte interface

- Une interface n'est pas une image statique mais un ensemble de composants graphiques, qui peuvent être des boutons, du texte, mais aussi des groupements d'autres composants graphiques, pour lesquels nous pouvons définir des attributs communs (taille, couleur, positionnement, etc.). Ainsi, l'écran ci-après (figure 3-1) peut-il être vu par le développeur comme un assemblage (figure 3-2).
- La représentation effective par le développeur de l'ensemble des composants graphiques se fait sous forme d'arbre, en une structure hiérarchique (figure 3-3). Il peut n'y avoir qu'un composant, comme des milliers, selon l'interface que vous souhaitez représenter.
- Dans l'arbre ci-après (figure 3-3), par exemple, les composants ont été organisés en 3 parties (*haut*, *milieu* et *bas*).

Les interfaces utilisateur

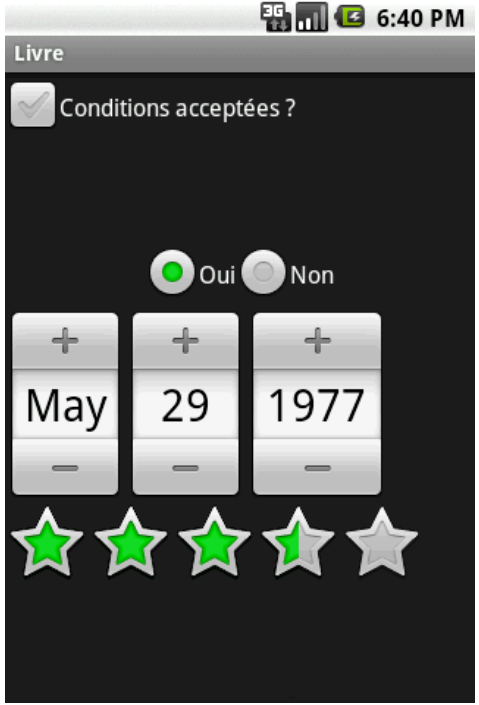


Figure 3-1

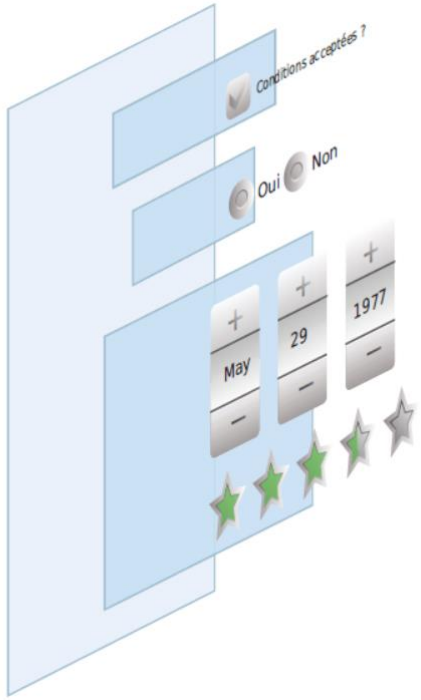


Figure 3-2

Les interfaces utilisateur

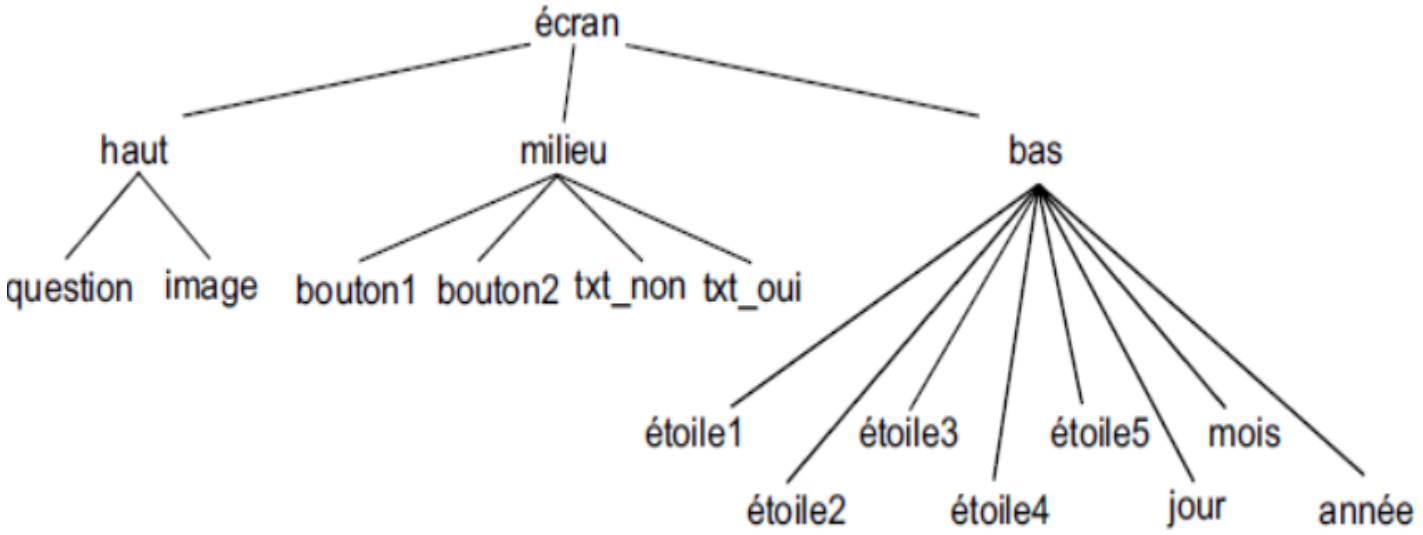


Figure 3-3

Les interfaces utilisateur

- Cet exemple donne l'idée générale de l'organisation des interfaces – car les données graphiques ne sont pas exactement représentées ainsi

Les interfaces utilisateur

Les vues

- Le composant graphique élémentaire de la plate-forme Android est la *vue* : tous les composants graphiques (boutons, images, cases à cocher, etc.) d'Android héritent de la classe View.
- Tout comme nous l'avions fait dans l'exemple précédent, Android vous offre la possibilité de regrouper plusieurs vues dans une structure arborescente à l'aide de la classe ViewGroup.
- Cette structure peut à son tour regrouper d'autres éléments de la classe ViewGroup et être ainsi constituée de plusieurs niveaux d'arborescence.
- L'utilisation et le positionnement des vues dans une activité se fera la plupart du temps en utilisant une mise en page qui sera composée par un ou plusieurs gabarits de vues

Les interfaces utilisateur

Positionner les vues avec les gabarits

- Un gabarit, ou *layout* dans la documentation officielle, ou encore mise en page, est une extension de la classe ViewGroup

- Vous pouvez imbriquer des gabarits les uns dans les autres, ce qui vous permettra de
- créer des mises en forme évoluées

- Vous pouvez utiliser différents types de gabarits. En fonction du type choisi, les vues et les gabarits seront disposés différemment :

Les interfaces utilisateur

Positionner les vues avec les gabarits

- **LinearLayout** : permet d'aligner de gauche à droite ou de haut en bas les éléments qui y seront incorporés. En modifiant la propriété orientation vous pourrez signaler au gabarit dans quel sens afficher ses enfants : avec la valeur horizontal, l'affichage sera de gauche à droite alors que la valeur vertical affichera de haut en bas

- **RelativeLayout** : ses enfants sont positionnés les uns par rapport aux autres, le premier enfant servant de référence aux autres .

- **FrameLayout** : c'est le plus basique des gabarits. Chaque enfant est positionné dans le coin en haut à gauche de l'écran et affiché par-dessus les enfants précédents, les cachant en partie ou complètement. Ce gabarit est principalement utilisé pour l'affichage d'un élément (par exemple, un cadre dans lequel on veut charger des images) ;

Les interfaces utilisateur

Positionner les vues avec les gabarits

- **TableLayout** : permet de positionner vos vues en lignes et colonnes à l'instar d'un tableau.

- Voici un exemple de définition déclarative en XML d'une interface contenant un
- gabarit linéaire (le gabarit le plus commun dans les interfaces Android).

```
<!--Mon primier gabarit uvs-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingVie..."
    tools:context="com.dieyeline.android.mobilebanque.MainActivity"
    tools:showIn="@layout/activity_main2">
</LinearLayout>
```

Les interfaces utilisateur

Gerer les evenements

- Sous Android, toutes les actions de l'utilisateur sont perçues comme un événement, que ce soit le clic sur un bouton d'une interface, le maintien du clic, l'effleurement d'un élément de l'interface, etc.

- Ces événements peuvent être interceptés par les éléments de votre interface pour exécuter des actions en conséquence.

- Le mécanisme d'interception repose sur la notion d'écouteurs, aussi appelés *listeners* la documentation Java. Il permet d'associer un événement à une méthode à appeler en cas d'apparition de cet événement

Les interfaces utilisateur

Gerer les evenements

- pourra définir un écouteur sur l'événement clic d'un bouton pour afficher un message « Bouton cliqué !». C'est justement ce que nous allons faire ci-après.

- Notez que les événements qui peuvent être interceptés ainsi que les méthodes associées sont imposés. Pour un événement OnClick (élément cliqué), la méthode associée sera OnClick()

- Avant de gérer un événement du type « cliquer sur un bouton », commençons par créer un gabarit avec un bouton centré au milieu de l'écran. Pour intégrer un bouton dans notre gabarit, il suffit d'ajouter une vue Button.

Les interfaces utilisateur

Gerer les evenements

- Modifiez les déclarations XML du fichier main.xml de l'exemple précédent pour y insérer un bouton.

```
<?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"
>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/monBouton"
android:text="Cliquez ici !« >
</Button>
</LinearLayout>
```

Les interfaces utilisateur

Gerer les evenements

- Une fois l'instance du bouton monBouton récupérée dans le code avec la référence R.id.monBouton, vous pouvez associer l'écouteur correspondant à l'événement désiré (ici, à l'aide de setOnClickListener.).

- Création d'un écouteur sur un bouton

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
```

Les interfaces utilisateur

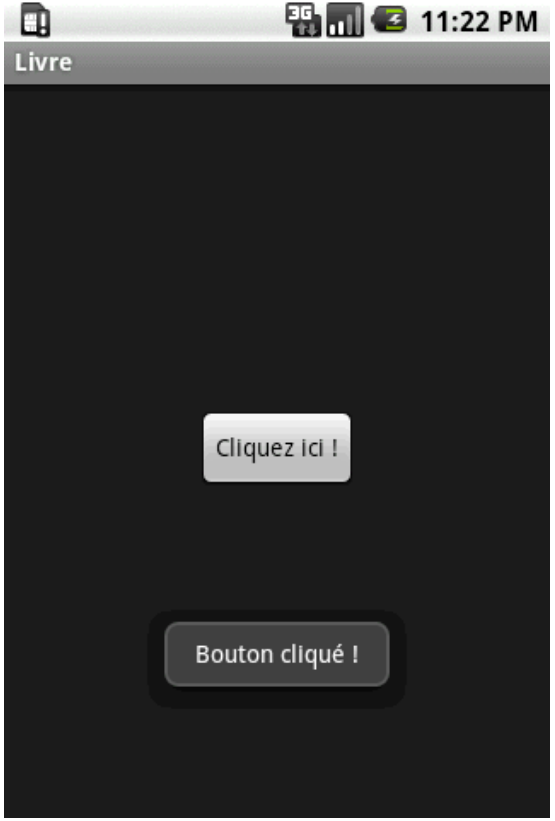
Gerer les evenements

```
import android.widget.Button;
import android.widget.Toast;

public class Main extends Activity {

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      setContentView(R.layout.main);

      // Nous cherchons le bouton dans notre interface
      ((Button)findViewById(R.id.monBouton))
      // Nous paramétrons un écouteur sur l'événement 'click' de ce bouton
      .setOnClickListener(new OnClickListener() {
         @Override
         public void onClick(View v) {
            // Nous affichons un message à l'utilisateur
            Toast.makeText(Main.this, "Bouton cliqué !", Toast.LENGTH_LONG).show();
         }
      });
   }
}
```

Les interfaces utilisateur

Gerer les evements

- Dans la méthode onClick de l'écouteur nous avons redéfinie, nous déclenchonsune alerte visuelle par l'intermédiaire d'un toast

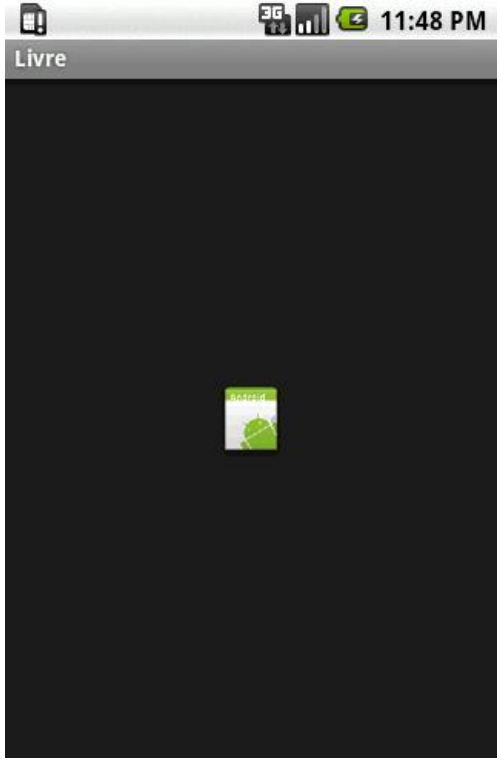


Les interfaces utilisateur

Intégrer les éléments graphiques dans votre interface

- Ajouter une image

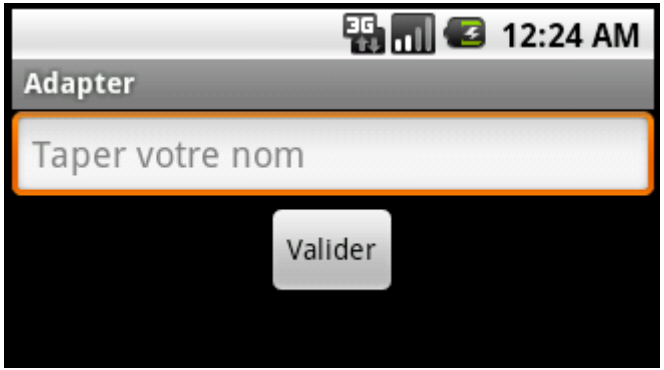
Pour ajouter une image dans votre interface, utilisez la vue de type ImageView. Vous pouvez spécifier la source de l'image directement dans votre définition XML, via l'attribut src, alors utiliser la méthode setImageResource en passant l'identifiant en paramètre.



Les interfaces utilisateur

Intégrer les éléments graphiques dans votre interface

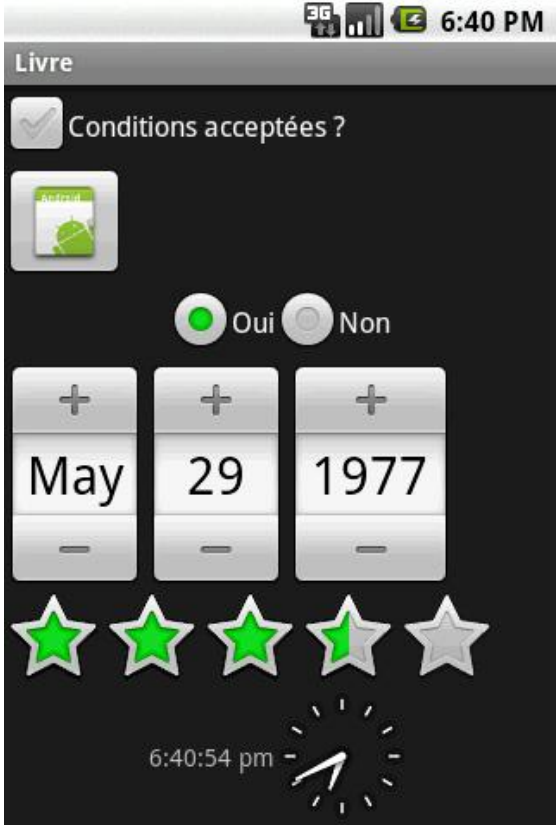
- Intégrer une boîte de saisie de texte
- Pour proposer à l'utilisateur de saisir du texte, vous pouvez utiliser la vue EditText. L'exemple suivant montre l'utilisation de cette vue. Modifiez le contenu du fichier main.xml de votre projet pour y insérer la déclaration de la vue EditText.



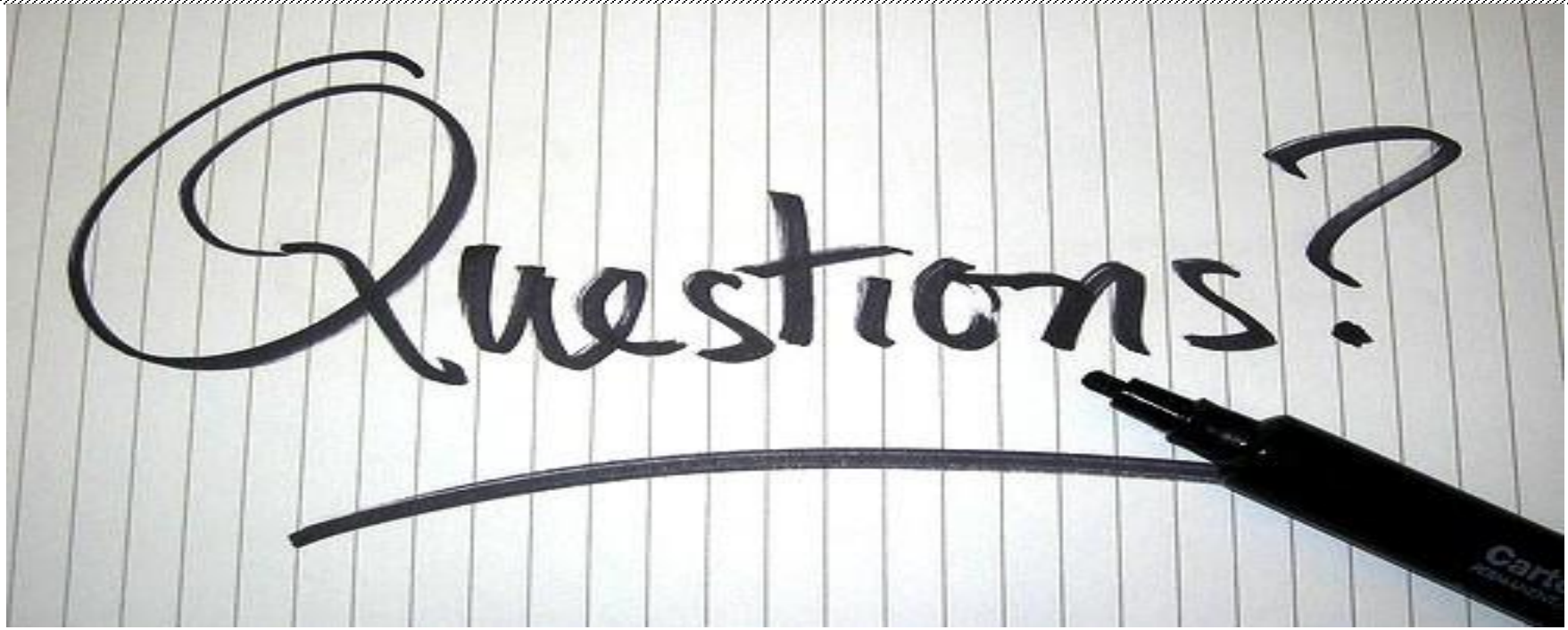
Les interfaces utilisateur

Intégrer les éléments graphiques dans votre interface

- Intégrer d'autres composants graphiques
- Dans l'exemple que nous vous proposons, allons ajouter les éléments suivants :une case à cocher (CheckBox), bouton image (ImageButton), deux boutons radio (RadioGroup et RadioButton), un contrôle de saisie de date (DatePicker une barre de vote (RatingBar) et deux horloges, l'une digitale (DigitalClock) et l'autre analogique (AnalogClock). Avant de vous expliquer comment fonctionne chacun ces éléments, voyons de façon concrète comment les intégrer dans votre application.



Questions



Bass.thiao@gmail.com