

**UVS**  
UNIVERSITE VIRTUELLE DU SENEGAL

android 



**DEVELOPPEMENT MOBILE ANDROID**

**Element d' Interaction et Groupes**

INSA BADJI Doctorant à l'Université de Thiès / Tuteur à



# Séquence 2: les heures, dates et fonts, Traitements des Evènements

- Objectif: Connaitre les éléments d'inter et groupe d'Android
- Plan
  - Dates et heures
  - Evènement Généraux et spécifique
  - Fonts



# Choix de date et d'heure

- DatePicker
  - `android:startYear` Pour définir l'année de départ du calendrier affiché
  - `android:endYear` Pour définir l'année de fin du calendrier affiché
  - `android:minDate` Pour définir la date affichée de départ du calendrier sous la forme mm/jj/aaaa
  - `android:maxDate` Pour définir la date affichée de fin du calendrier sous la forme mm/jj/aaaa
- TimePicker

# Choix de date et d'heure: exemple

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
    <TextView android:id="@+id/dateAndTime"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
    <Button android:id="@+id/dateBtn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text=" Régler la date"
    />
    <Button android:id="@+id/timeBtn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text=" Régler l'heure"
    />
</LinearLayout>
```



# Choix de date et d'heure: exemple

```
public class ChronoDemo extends Activity {
    DateFormat fmtDateAndTime=DateFormat.getDateTimeInstance();
    TextView dateAndTimeLabel;
    Calendar dateAndTime=Calendar.getInstance();
    DatePickerDialog.OnDateSetListener d=new DatePickerDialog.OnDateSetListener() {
        public void onDateSet(DatePicker view, int year, int monthOfYear,
            int dayOfMonth) { dateAndTime.set(Calendar.YEAR,
                year); dateAndTime.set(Calendar.MONTH, monthOfYear);
                dateAndTime.set(Calendar.DAY_OF_MONTH, dayOfMonth);
                updateLabel();
            }
    };
    TimePickerDialog.OnTimeSetListener t=new TimePickerDialog.OnTimeSetListener() {
        public void onTimeSet(TimePicker view, int hourOfDay,
            int minute) {
            dateAndTime.set(Calendar.HOUR_OF_DAY, hourOfDay);
            dateAndTime.set(Calendar.MINUTE, minute);
            updateLabel();
        }
    };
};
```

# Choix de date et d'heure: exemple

```
@Override
public void onCreate(Bundle icle) {
    super.onCreate(icle);
    setContentView(R.layout.main);

    Button btn=(Button)findViewById(R.id.dateBtn);

    btn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            new DatePickerDialog(ChronoDemo.this,
                d,
                dateAndTime.get(Calendar.YEAR),
                dateAndTime.get(Calendar.MONTH),
                dateAndTime.get(Calendar.DAY_OF_MONTH)).show();
        }
    });

    btn=(Button)findViewById(R.id.timeBtn);

    btn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            new TimePickerDialog(ChronoDemo.this, t,
                dateAndTime.get(Calendar.HOUR_OF_DAY),
                dateAndTime.get(Calendar.MINUTE),
                true).show();
        }
    });

    dateAndTimeLabel=(TextView)findViewById(R.id.dateAndTime);

    updateLabel();
}

private void updateLabel() {
    dateAndTimeLabel.setText(fmtDateAndTime
        .format(dateAndTime.getTime()));
}
}
```

# ProgressBar

- Deux comportements selon que l'on connaît ou pas la valeur maximale: `android:indeterminate` Pour définir le type de progressBar (true=indéterminé, false=déterminé).
- Animation (si indéterminé): `android:indeterminateBehavior="i"` (où i peut être : `repeat` ou `cycle`) définit le comportement de l'animation pour le type circulaire (repeat=recommence l'animation, cycle=changer le sens de l'animation)
- Dimensions
  - `android:maxHeight="unité"`
  - `android:minHeight="unité"`
  - `android:maxWidth="unité"`
  - `android:minWidth="unité"`
- Valeurs (si déterminé)
  - `android:max` Pour définir la valeur maximale
  - `android:progress` Pour définir la valeur initiale
  - `android:secondaryProgress` Pour définir une valeur secondaire (par exemple celle d'un buffer comme on le voit sur des vidéos en streaming)



# Formes des ProgressBar

- En l'absence de paramètre `style` la forme est circulaire
- Pour obtenir d'autres forme on utilise le paramètre `style` :
  - `style="?android:attr/s"` où s peut être :
    - `progressBarStyleHorizontal`
    - `progressBarStyleSmall`
    - `progressBarStyleLarge`



- On ne peut pas changer la couleur

## SeekBar



C'est un ProgressBar sous forme de barre horizontale dotée d'un curseur permettant de modifier la valeur si on a choisi

`android:indeterminate="false"`

sinon le curseur ne marche pas et la barre bouge sans arrêt.



# Example ProgressBar

```
<ProgressBar android:id="@+id/progres"  
  android:layout_width="wrap_content"  
  android:layout_height="wrap_content"  
  android:minWidth="300px"  
  android:minHeight="30px"  
  android:max="100"  
  android:progress="30"  
  android:secondaryProgress="40"  
  android:indeterminate="false"  
  style="?android:attr/progressBarStyleHorizontal"/>
```

# RatingBar



Paramètres :

[android:isIndicator](#) Pour indiquer si l'utilisateur peut modifier la valeur ou pas (true= non modifiable)

[android:numStars](#) Pour définir le nombre d'étoiles affichées

[android:rating](#) Pour définir la position initiale

[android:stepSize](#) Pour définir le pas de progression (on peut colorier des  $\frac{1}{4}$  d'étoiles par exemple)

# Horloges et Chronomètres

AnalogClock



DigitalClock

4:58:17 pm

Chronometer

Ce cours a commencé depuis 51:45 déjà

[android:format="f"](#) (où f est une chaîne dans laquelle la première occurrence de %s sera remplacée par la valeur du chronomètre sous la forme MM:SS ou H:MM:SS)

# AnalogClock

```
<AnalogClock android:id="@+id/horloge"  
  android:layout_width="wrap_content"  
  android:layout_height="wrap_content"  
/>
```

# Sélection Date/heure: Date

- Android supporte également les widgets (**DatePicker**, **TimePicker**) et boîtes de dialogue (**DatePickerDialog**, **TimePickerDialog**) pour aider les utilisateurs entrer des dates et des heures.
- Le **DatePicker** et **DatePickerDialog** permettent de régler la date de début de sélection, sous la forme d'une année, mois et jour. Valeur du mois traverse de 0 pour janvier 11 pour le 2 décembre décembre.
- Chaque widget fournit un objet de rappel **OnDateChangeListener** ou **OnDateSetListener**) où vous êtes informé d'une nouvelle date sélectionnée par l'utilisateur.


# Sélection Date/heure: Heure

**TimePicker** et **TimePickerDialog** permettent de:

1. régler le temps initial, l'utilisateur peut ajuster, sous la forme d'une **heure** (**0** à **23**) et une **minute** (**0** à **59**)
2. *indiquer si la sélection doit être en **mode 12-heure** (AM/PM), ou **mode 24-heure**.*
3. fournir un objet de rappel (**OnTimeChangeListener** or **OnTimeSetListener**) pour être averti des lors que l'utilisateur a choisi une nouvelle fois (ce qui vous est fournie sous la forme d'une heure et minute)

# Sélection Date/heure: exemple

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/widget28"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    <TextView
        android:id="@+id/lblDateAndTime"
        android:layout_width="fill_parent"
        android:layout_height="47px"
        android:background="#ff000099"
        android:textStyle="bold"
    >
    </TextView>
    <Button
        android:id="@+id/btnDate"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Set the Date"
    >
    </Button>
    <Button
        android:id="@+id/btnTime"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Set the Time"
    >
    </Button>
</LinearLayout>
```





# Sélection Date/heure: exemple

```
package cis493.demoui;
import android.app.Activity;
import android.os.Bundle;
import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TimePicker;
import android.widget.TextView;
import java.text.DateFormat;
import java.util.Calendar;
```

```
public class AndDemoUI extends Activity {
    DateFormat fmtDateAndTime = DateFormat.getDateTime();
    TextView lblDateAndTime;
    Calendar myCalendar = Calendar.getInstance();
```

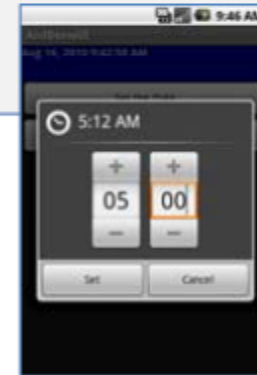


```
    DatePickerDialog.OnDateSetListener d = new DatePickerDialog.OnDateSetListener()
    {
        public void onDateSet(DatePicker view,
                               int year, int monthOfYear, int dayOfMonth) {
            myCalendar.set(Calendar.YEAR, year);
            myCalendar.set(Calendar.MONTH, monthOfYear);
            myCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
            updateLabel();
        }
    };
```

# Sélection Date/heure: exemple

```
TimePickerDialog.OnTimeSetListener t = new TimePickerDialog.OnTimeSetListener()
{
    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        myCalendar.set(Calendar.HOUR_OF_DAY, hourOfDay);
        myCalendar.set(Calendar.MINUTE, minute);
        updateLabel();
    }
};
```

```
private void updateLabel() {
    lblDateAndTime.setText(fmtDateAndTime.format(myCalendar.getTime()));
}
```



# Sélection Date/heure: exemple

```
@Override
public void onCreate(Bundle icle) {
    super.onCreate(icle);
    setContentView(R.layout.main);
    lblDateAndTime = (TextView) findViewById(R.id.lblDateAndTime);
    Button btnDate = (Button) findViewById(R.id.btnDate);
    btnDate.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            new DatePickerDialog(AndDemoUI.this, d,
                                myCalendar.get(Calendar.YEAR),
                                myCalendar.get(Calendar.MONTH),
                                myCalendar.get(Calendar.DAY_OF_MONTH)).show();
        }
    });

    Button btnTime = (Button) findViewById(R.id.btnTime);
    btnTime.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            new TimePickerDialog(AndDemoUI.this, t,
                                myCalendar.get(Calendar.HOUR_OF_DAY),
                                myCalendar.get(Calendar.MINUTE), true).show();
        }
    });

    updateLabel();
} // onCreate
} // class
```

# Onglets

- Une Activity peut être subdivisée en plusieurs onglets
- Afin d'utiliser des onglets, il vous faut les prochaines pièces:
  - TabHost : le conteneur surplombant les boutons d'onglets et leur contenus
  - TabWidget : la rangée de boutons d'onglets qui contient le texte et optionnellement les icones
  - FrameLayout : le conteneur pour les contenus des onglets, chaque onglet est un enfant du FrameLayout

# Onglets : quelques règles

- Afin que les onglets fonctionnent correctement, il faut suivre certaines règles:
- Le TabWidget doit avoir une **android:id** qui est [@android:id/tabs](#)
- Il est conseillé de mettre du padding entre le FrameLayout et les tab buttons, afin que le contenu ne «colle» pas aux tab buttons.
- Il est possible d'utiliser une TabActivity, mais il faut alors donner au TabHost une **android:id** of [@android:id/tabhost](#)
- TabActivity regroupe de la fonctionnalité propre aux onglets et facilite l'usage d'onglets

# Exemple de layout avec onglets

```
<TabHost android:id="@+id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TabWidget android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
        />
        <FrameLayout android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent">
            <AnalogClock android:id="@+id/tab1"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_centerHorizontal="true"
            />
            <Button android:id="@+id/tab2"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:text="A semi-random button"
            />
        </FrameLayout>
    </LinearLayout>
</TabHost>
```

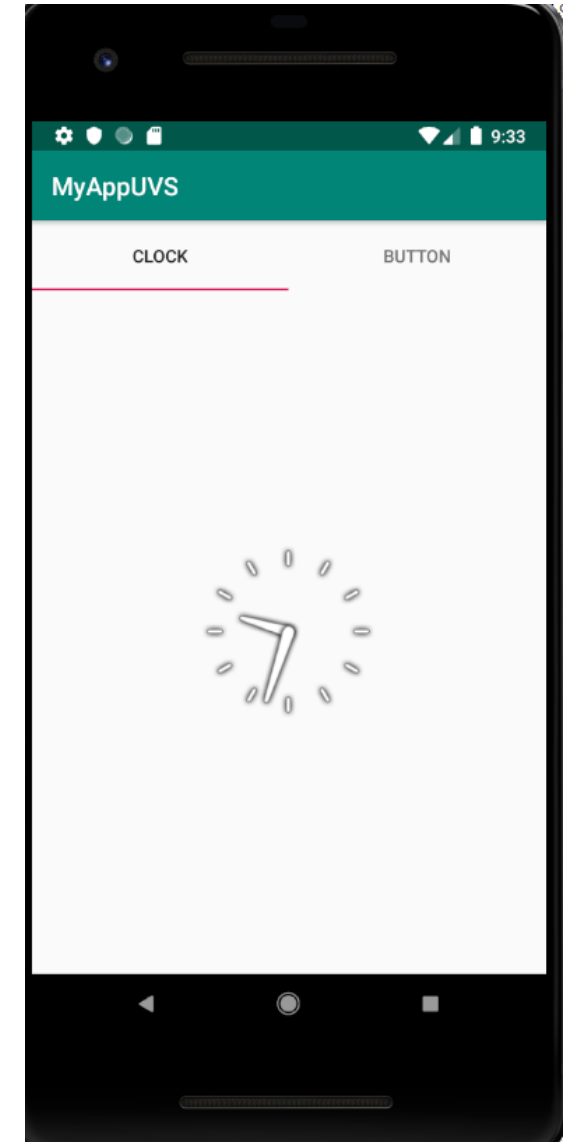
# Onglets : comment faire

- Il est impératif d'utiliser du code Java afin de dire au **TabHost** **quelle view** représente le contenu d'un bouton mais aussi à quoi ressemblera un bouton
- Pour ceci, nous utilisons un objet **TabSpec**: en utilisant la fonction **newTabSpec()** sur le **TabHost**
- Il y a ensuite deux fonctions clés:
  - **setContent()** pour indiquer le contenu d'un onglet
    - android:id de la view
    - ou un Intent (contenu d'une autre Activity, par exemple)
  - **setIndicator()**
    - texte pour l'onglet
    - image drawable pour l'icone
    - view
- When tabs are ready, call the setup() method on the tabhost and add the tabs



# Onglets : exemple d'activity

```
public class TabDemo extends Activity {  
    @Override  
    public void onCreate(Bundle icicle) {  
        super.onCreate(icicle);  
        setContentView(R.layout.main);  
        TabHost tabs=(TabHost)findViewById(R.id.tabhost);  
  
        tabs.setup();  
  
        TabHost.TabSpec spec=tabs.newTabSpec("tag1");  
  
        spec.setContent(R.id.tab1);  
        spec.setIndicator("Clock");  
        tabs.addTab(spec);  
  
        spec=tabs.newTabSpec("tag2");  
        spec.setContent(R.id.tab2);  
        spec.setIndicator("Button");  
        tabs.addTab(spec);  
    }  
}
```



# Traitement des événements

- Tous les éléments d'interface (conteneurs et widgets) possèdent les méthodes suivantes :
  - `setOnClickListener(View.OnClickListener)` associe un écouteur d'événements aux **clics** sur la vue
  - `setOnLongClickListener(View.OnLongClickListener)` associe un écouteur d'événements aux **clics longs** sur la vue
  - `setKeyListener(View.OnKeyListener)` associe un écouteur d'événements aux **actions clavier** sur la vue
  - `setOnTouchListener(View.OnTouchListener)` associe un écouteur d'événements aux **touchés** sur la vue
- qui permettent de leur associer des écouteurs d'événements
- Certains éléments ont des écouteurs spécifiques

# Traitement des événements (les bonnes habitudes)

- Quand un widget est modifié la méthode correspondante de l'écouteur d'événements associé est exécutée
- Ceci est vrai que le widget soit modifié par l'utilisateur ou par programme.
- Il est donc préférable de ne mettre en place les écouteurs d'événements qu'après avoir totalement initialisé les widgets pour éviter qu'ils ne s'exécutent au cours de ces initialisations

# Evénements généraux

Evénement	Association Classe	Méthode • Paramètres
Clic	setOnClickListener <a href="#">View.OnClickListener</a>	<a href="#">onClick(View)</a> • Élément concerné
Clic long	setOnLongClickListener <a href="#">View.OnLongClickListener</a>	<a href="#">onLongClick(View)</a> • Élément concerné
Clavier	setOnKeyListener <a href="#">View.OnKeyListener</a>	<a href="#">onKey(View, int, KeyEvent)</a> • Élément concerné • Code clavier • Evénement clavier
Touché	setOnTouchListener <a href="#">View.OnTouchListener</a>	<a href="#">onTouch(View, MotionEvent)</a> • Élément concerné • Evénement de touché

# Evénements généraux

- ListView , GridView et Gallery

Evénement sur un élément	Association Classe	Méthode • Paramètres
Clic	setOnItemClickListener <a href="#">AdapterView.OnItemClickListener</a>	<a href="#">onItemClick</a> ( <a href="#">AdapterView</a> , <a href="#">View</a> , int, long) <ul style="list-style-type: none"><li>• Gestionnaire de contenu</li><li>• Élément concerné</li><li>• Rang de l'élément</li><li>• Identifiant de l'élément</li></ul>
Clic long	setOnItemLongClickListener <a href="#">AdapterView.OnItemLongClickListener</a>	<a href="#">onItemLongClick</a> ( <a href="#">AdapterView</a> , <a href="#">View</a> , int, long) <ul style="list-style-type: none"><li>• Idem</li></ul>
Sélection	setOnItemSelectedListener <a href="#">AdapterView.OnItemSelectedListener</a>	<a href="#">onItemSelected</a> ( <a href="#">AdapterView</a> , <a href="#">View</a> , int, long) <ul style="list-style-type: none"><li>• Idem</li></ul> <a href="#">onNothingSelected</a> ( <a href="#">AdapterView</a> )

# Evénements spécifiques

- Spinner et AutoCompleteTextView

Evénement sur un élément	Association Classe	Méthode • Paramètres
Sélection	setOnItemSelectedListener <a href="#">AdapterView.OnItemSelectedListener</a>	<a href="#">onItemSelected</a> ( <a href="#">AdapterView</a> , <a href="#">View</a> , int, long) <ul style="list-style-type: none"><li>• Élément permettant le choix</li><li>• Élément concerné</li><li>• Rang de l'élément</li><li>• Identifiant de l'élément</li></ul> <a href="#">onNothingSelected</a> ( <a href="#">AdapterView</a> )

# Evénements spécifiques

- TextView et EditText

Evénement	Association Classe	Méthode • Paramètres
Fin de saisie	setOnEditorActionListener <a href="#">TextWatcher</a>	<a href="#">onEditorAction</a> ( <a href="#">TextView</a> , int, <a href="#">KeyEvent</a> ) <ul style="list-style-type: none"><li>• Elément concerné</li><li>• EditorInfo.IME_NULL (si touche Entrée)</li><li>• Evénement clavier (si touche Entrée)</li></ul>
Modification	addTextChangedListener <a href="#">TextChangedListener</a>	<a href="#">beforeTextChanged</a> ( <a href="#">CharSequence</a> , int, int, int) <a href="#">afterTextChanged</a> ( <a href="#">CharSequence</a> , int, int, int) <ul style="list-style-type: none"><li>• Texte</li><li>• Point de départ de la modification</li><li>• Nombre de cars remplacés</li><li>• Nombre de cars de remplacement</li></ul>
Saisie	setKeyListener <a href="#">KeyListener</a>	<a href="#">onKeyDown</a> ( <a href="#">View</a> , <a href="#">Editable</a> , int, <a href="#">KeyEvent</a> ) <a href="#">onKeyUp</a> ( <a href="#">View</a> , <a href="#">Editable</a> , int, <a href="#">KeyEvent</a> ) <ul style="list-style-type: none"><li>• Elément concerné</li><li>• Texte</li><li>• Code de la touche</li><li>• Evénement clavier</li></ul>



# Evénements spécifiques

- DatePicker

Événement de choix	Association Classe	Méthode • Paramètres
Choix	init <a href="#">DatePicker.OnDateChangeListener</a>	<a href="#">onDateChanged</a> ( <a href="#">DatePicker</a> , int, int, int) <ul style="list-style-type: none"><li>• Élément concerné</li><li>• Année</li><li>• Mois</li><li>• Jour</li></ul>

- TimePicker

Événement de choix	Association Classe	Méthode • Paramètres
Choix	setOnTimeChangeListener <a href="#">TimePicker.OnTimeChangeListener</a>	<a href="#">onTimeChanged</a> ( <a href="#">TimePicker</a> , int, int) <ul style="list-style-type: none"><li>• Élément concerné</li><li>• Heure</li><li>• Minutes</li></ul>

# Evénements spécifiques

- SeekBar

Evénement	Association Classe	Méthode • Paramètres
Curseur déplacé	setOnSeekBarChangeListener <a href="#">SeekBar.OnSeekBarChangeListener</a>	<a href="#">onProgressChanged( SeekBar, int, boolean)</a> <ul style="list-style-type: none"><li>• Élément concerné</li><li>• Position du curseur</li><li>• Action de l'utilisateur</li></ul>
Début de déplacement		<a href="#">onStartTrackingTouch( SeekBar)</a> <ul style="list-style-type: none"><li>• Élément concerné</li></ul>
Fin de déplacement		<a href="#">onStopTrackingTouch( SeekBar)</a> <ul style="list-style-type: none"><li>• Élément concerné</li></ul>

# Evénements spécifiques

- RatingBar

Evénement	Association Classe	Méthode • Paramètres
Valeur modifiée	setOnRatingBarChangeListener <a href="#">RatingBar.OnRatingBarChangeListener</a>	<a href="#">onRatingChanged(RatingBar, float, boolean)</a> <ul style="list-style-type: none"><li>• Élément concerné</li><li>• Valeur choisie</li><li>• Action de l'utilisateur</li></ul>

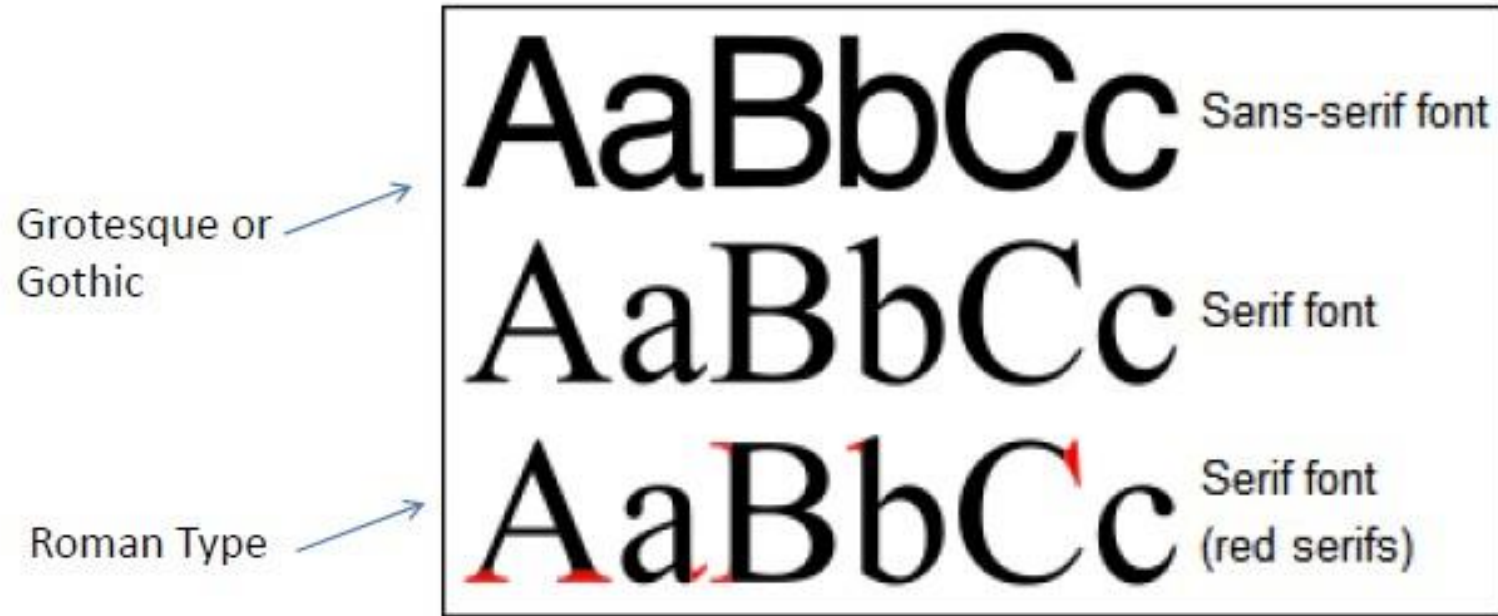
- Chronometer

Evénement	Association Classe	Méthode • Paramètres
Incrémentation	setOnChronometerTickListener <a href="#">Chronometer.OnChronometerTickListener</a>	<a href="#">onChronometerTick(Chronometer)</a> <ul style="list-style-type: none"><li>• Élément concerné</li></ul>

# Les Fonts

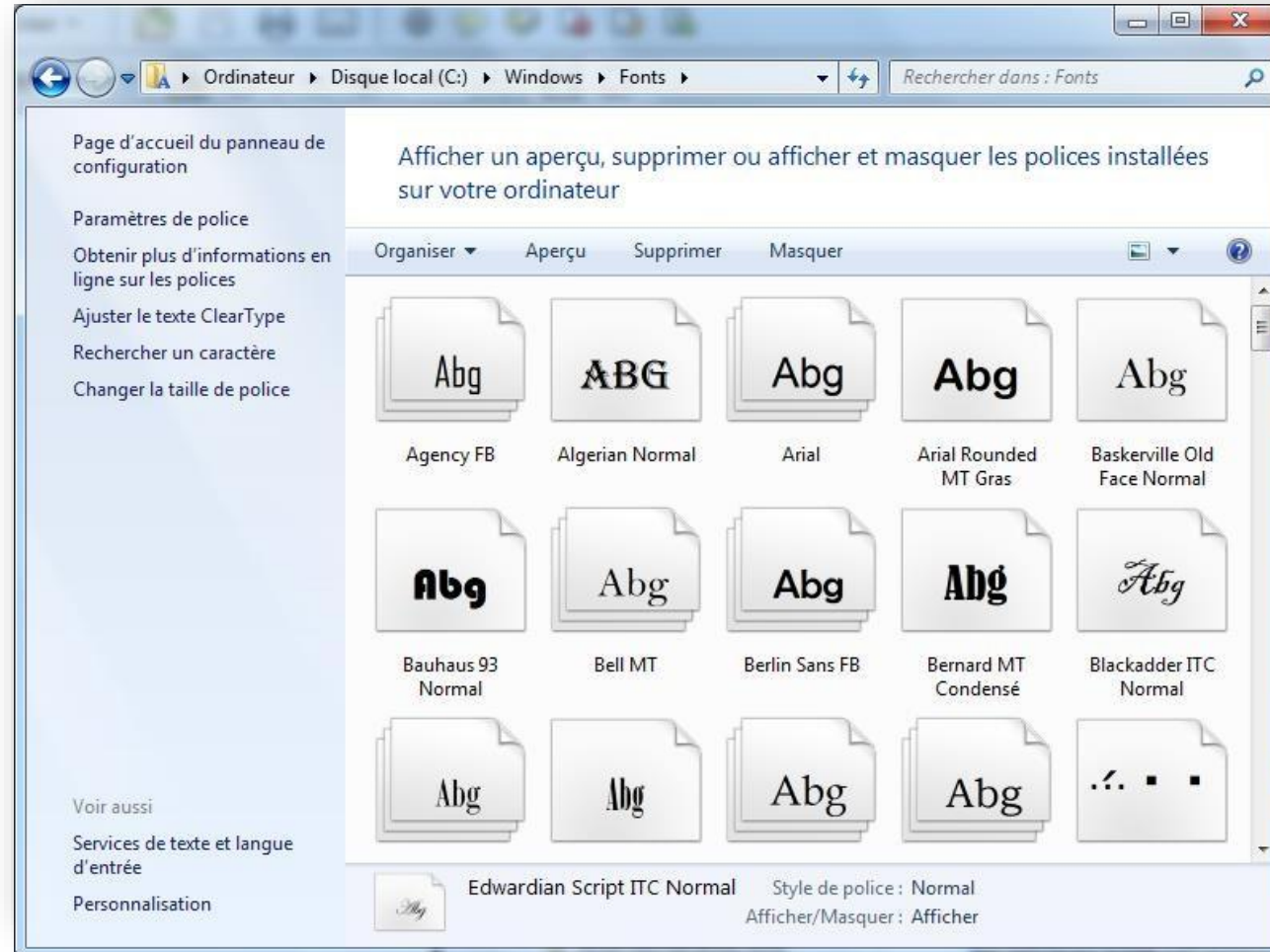
- Android propose *naturellement* trois familles de fonts:
  - **sans**: Une étudiante fantôme en CFPP
  - **serif**: Une étudiante fantôme en CFPP
  - **monospaced**: Une étudiante fantôme en CFPP

# Fonts



Source: <http://en.wikipedia.org/wiki/Serif>

# Les autres Fonts(Windows)



# Fonts

Pour ajouter une nouvelle famille de font à votre application:

1. Créer un répertoire **/fonts** dans **/assets**
2. Copier *la ou les fonts* que vous souhaitez utiliser dans ce répertoire.
3. Utiliser java pour appliquer la police d'écriture(Voir exemple)



**Fin de la  
séquence 3**