

# PROGRAMMATION ANDROID



1

Ahcène Bounceur  
Université de Bretagne Occidentale

# *Chapitre 2*

A decorative graphic on the left side of the slide. It features a series of vertical lines in various shades of green and grey. Overlaid on these lines are several green circles of different sizes. One circle is particularly large and contains the number '2'.

2

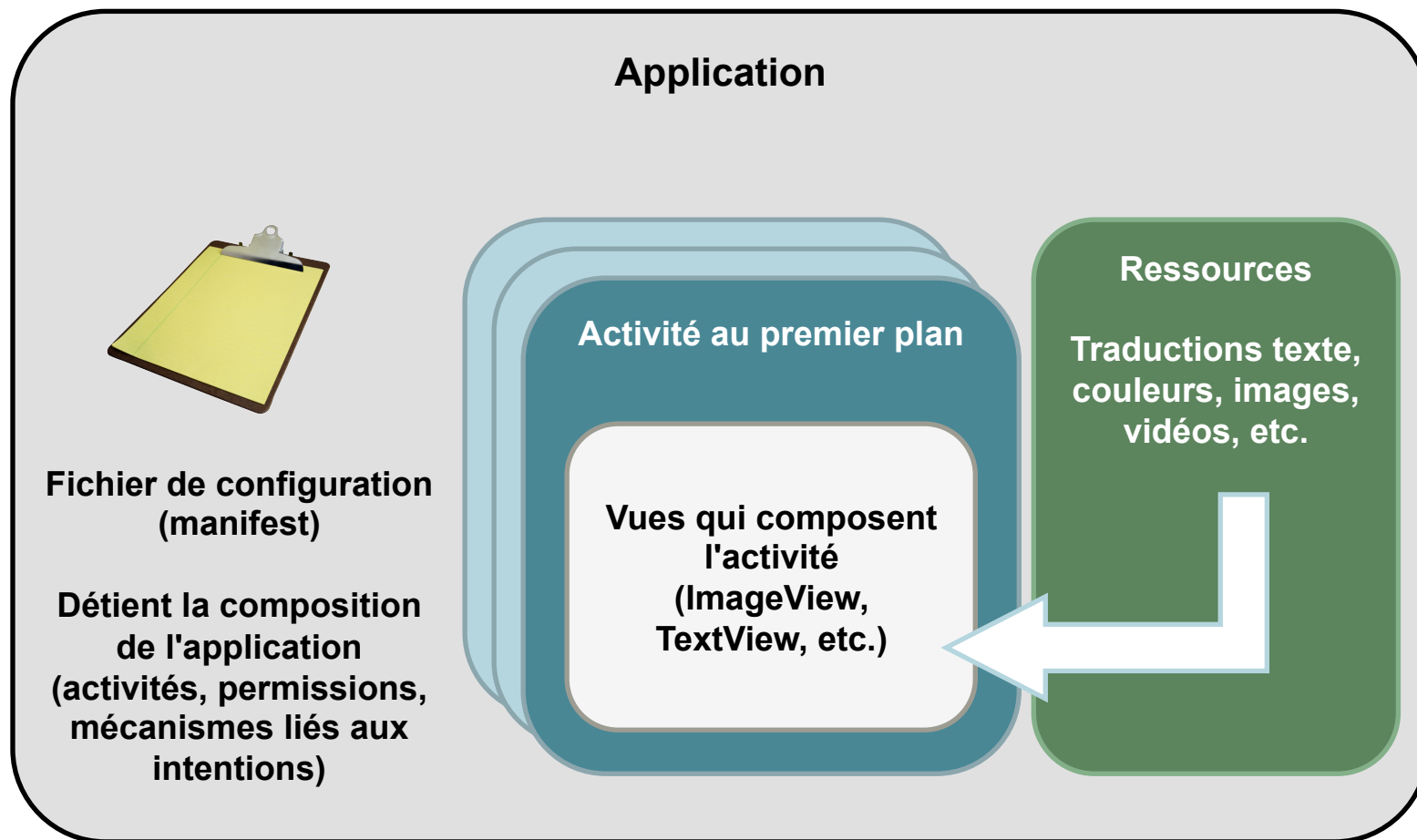
Création d'applications et les Activités

***Ce chapitre vous présentera les  
différents composants d'une  
application Android et leurs  
interactions.***

# APPLICATION ANDROID

- Une application Android est un assemblage de composants liés grâce à un fichier de configuration.
  - Les **Activités**
    - Classe Java
  - Les **Vues et contrôles**
    - Fichiers XML ou code Java
  - Les **Ressources**
    - Interfaces (Layout), Images, Textes, audio, vidéo, ...
  - Le **fichier de configuration**
    - AndroidManifest.xml

# Création d'applications Android (concepts)



# LES PIÈCES D'UNE APPLICATION ANDROID

- Une application Android est composée de plusieurs éléments qu'il faut assembler pour obtenir un tout cohérent. Plus une application est complexe, plus le nombre de pièces utilisées sera grand. Voici donc les différents pièces principales de l'Android :

- Activités
- Services
- Fournisseurs de contenu
- Gadgets

Composants applicatifs

- Objets **Intent**
- Récepteurs d'Intents
- Notifications
- Filtres d'Intent

Éléments d'interaction

# COMPOSANTS APPLICATIFS

- Activité (`android.app.Activity`)
  - représente le bloc de base d'une application.
- Service (`android.app.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.
- Fournisseur de contenu (content provider) (`android.content.ContentProvider`)
  - permet de gérer et de partager des informations.
- Gadget (`Android.appwidget.*`)
  - est un composant graphique qui s'installe sur le bureau Android.

# ELÉMENTS D'INTERACTION

- Intents (`android.content.Intent`)
  - il permet de diffuser des messages en demandant la réalisation d'une action.
- Récepteurs (`android.content.BroadcastReceiver`)
  - 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.
- Notifications (`android.app.Notification`)
  - une notification signale une information à l'utilisateur sans interrompre ses actions en cours.



# ELÉMENTS D'INTERACTION

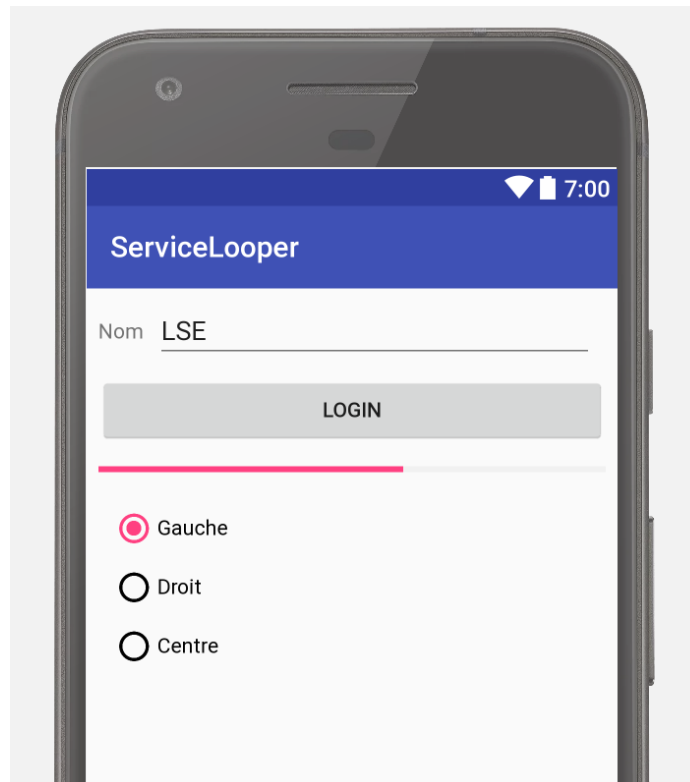
- Filtres (d'Intents) : un objet Intent peut mentionner explicitement un composant cible. Si cette information n'est pas renseignée, Android doit choisir le meilleur composant pour y répondre. Ceci est mené à bien via une comparaison de l'objet Intent avec les filtres des différentes applications cibles. Les filtres ne se manipulent généralement pas via l'API mais sont paramétrables grâce à la balise <intent-filter> du fichier de configuration.

# LE FICHIER DE CONFIGURATION

- Chaque application Android nécessite un fichier de configuration : **AndroidManifest.xml**
- Ce fichier est placé dans le répertoire de base du projet, à sa racine.
- Il décrit le contexte de l'application, les activités, les services, les récepteurs d'Intents (*Broadcast receivers*), les fournisseurs de contenu et les permissions.

# LES VUES

- Les vues sont les briques de construction de l'interface graphique d'une activité Android. Les objets **View** représentent des éléments à l'écran qui permettent d'interagir avec l'utilisateur via un mécanisme d'événements
- Un écran Android contient un arbre d'éléments de type **View** dont chaque élément est différent de par ses propriétés de forme, de taille, etc.



# LES RESSOURCES

- Les ressources sont des fichiers externes – ne contenant pas d'instruction – qui sont utilisés par le code et liés à votre application au moment de sa construction. Android offre un support d'un grand nombre de fichiers ressources comme les fichiers images JPEG et PNG, les fichiers XML, etc.
- Le fichier qui permet de créer le lien entre les ressources et le code Java est appelé R.java

# LES RESSOURCES

- Lien vers un layout :

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

# LES RESSOURCES

- Lien vers un composant graphique d'un layout :

```
Type nom = (Type) findViewById(R.id.nom_composant)
```

# LES RESSOURCES

- Lien vers strings.xml (simple String):

```
String s = getString(R.string.simpleString);
```

# LES RESSOURCES

- Lien vers strings.xml (String paramétré ou formaté) :

Dans **strings.xml** :

```
<string name="message">Monsieur %1$s (%2$d ans)</string>
```

Sans le code :

```
String s = String.format(getString(R.string.message),  
"Johnny", 40);
```

Du java tout simplement :

```
String s = String.format("Monsieur %1$s (%2$d ans)", "Johnny", 40);
```



# LES RESSOURCES

- Lien vers strings.xml (Tableau) :

Dans strings.xml :

```
<string-array name="montableau">  
    <item>Brest</item>  
    <item>Paris</item>  
    <item>Lille</item>  
</string-array>
```

Sans le code :

```
String [] t =  
getResources().getStringArray(R.array.montableau);
```

# LES RESSOURCES

- Lien vers strings.xml (gestion du pluriel) :

Dans strings.xml :

```
<plurals name="universite">  
    <item quantity="one">Une seule université.</item>  
    <item quantity="other">%d universités.</item>  
</plurals>
```

Sans le code :

```
String s =  
getResources().getQuantityString(R.plurals.universite, 2);
```

# LES RESSOURCES

- Lien vers une couleur :

```
int c = getColor(R.color.colorPrimary)
```

```
int c = getColor(R.color.seafoam)
```

Ressources (color.xml) :

```
<color name="seafoam">#ffc6f9e5</color>
```

# LES RESSOURCES

- Lien vers une image :

```
Drawable drawable =  
getResources().getDrawable(R.drawable.myimage);
```

```
Bitmap img =  
BitmapFactory.decodeResource(getResources(),  
R.drawable.monimage);
```

```
Bitmap img = BitmapFactory.decodeStream(stream);
```

# LES RESSOURCES

- Peuvent être utilisées dans les propriétés des composants graphiques directement dans l'interface dédiée :

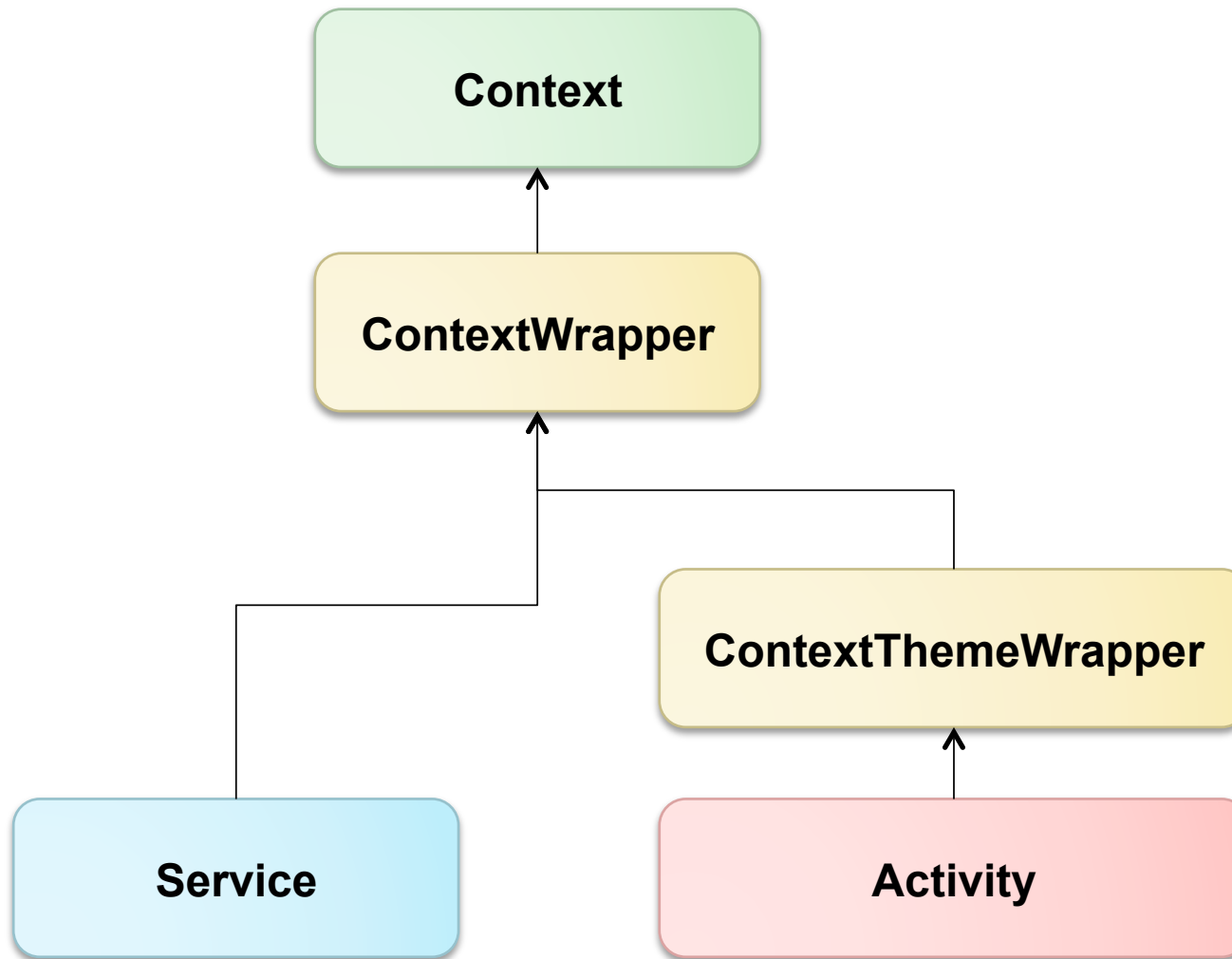
Pour un **textView**, on peut modifier sa propriété **text** liée à une ressource (strings.xml) comme suit :

**@string/nom\_string**

# ACTIVITÉ

- Une activité peut être assimilée à un écran qu'une application propose à son utilisateur.
- Chaque écran d'une application correspond à une activité
- Une activité est composée de deux volets :
  - la logique de l'activité et la gestion du cycle de vie de l'activité qui sont implémentés en Java dans une classe héritant de **Activity**,
  - l'interface utilisateur, qui pourra être définie soit dans le code de l'activité soit de façon plus générale dans un fichier XML placé dans les ressources de l'application.

# ACTIVITÉ



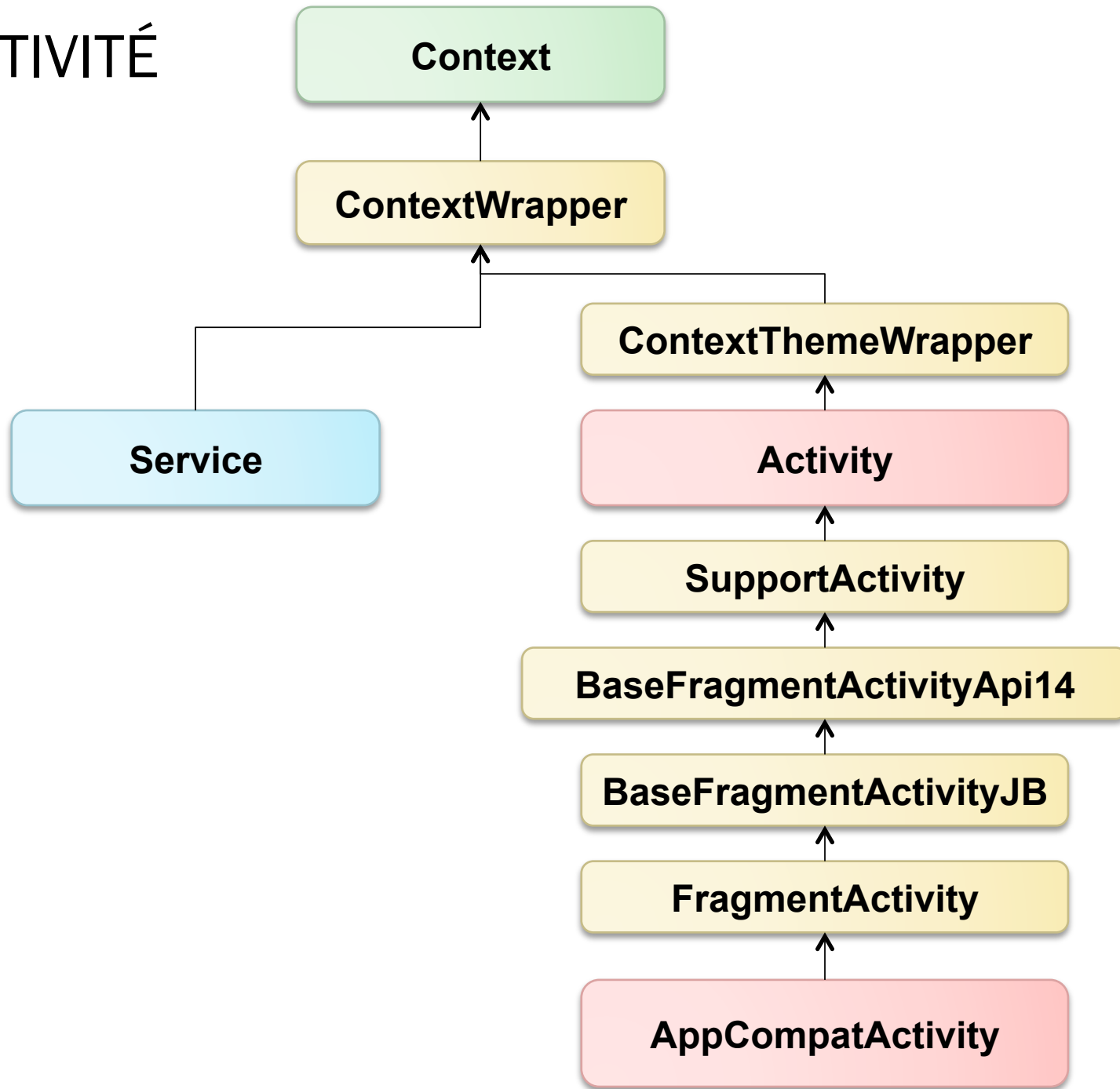
# ACTIVITÉ

```
import android.app.Activity;
import android.os.Bundle;

public class ActiviteSimple extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```



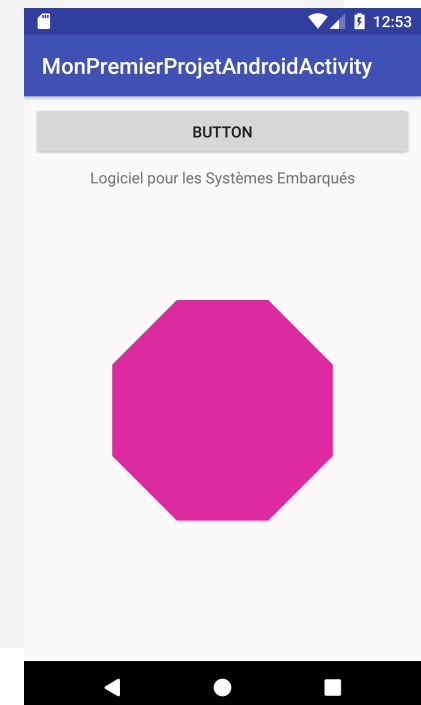
# ACTIVITÉ



# ACTIVITÉ (AppCompatActivity)

```
import android.app.Activity;
import android.os.Bundle;

public class ActiviteSimple extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```



# ACTIVITÉ (Plein écran)

```
import android.app.Activity;
import android.os.Bundle;

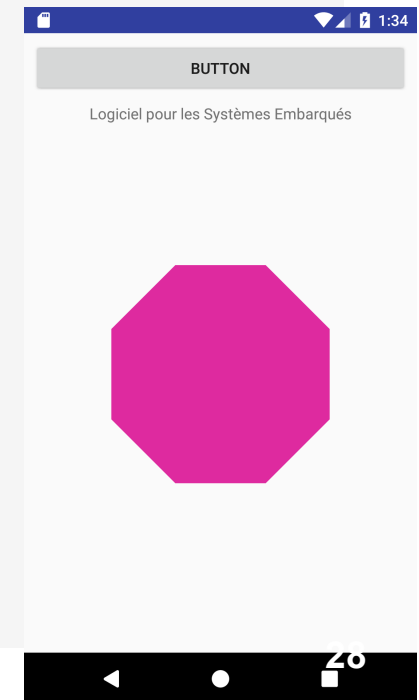
public class ActiviteSimple extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getWindow().setFlags(
            WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN
        );
    }
}
```



# ACTIVITÉ (Sans barre d'action)

```
import android.app.Activity;
import android.os.Bundle;

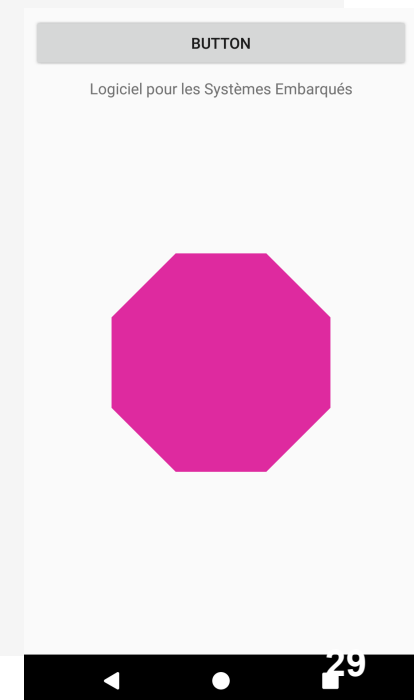
public class ActiviteSimple extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getSupportActionBar().hide();
    }
}
```



# ACTIVITÉ (Plein écran et Sans barre d'action)

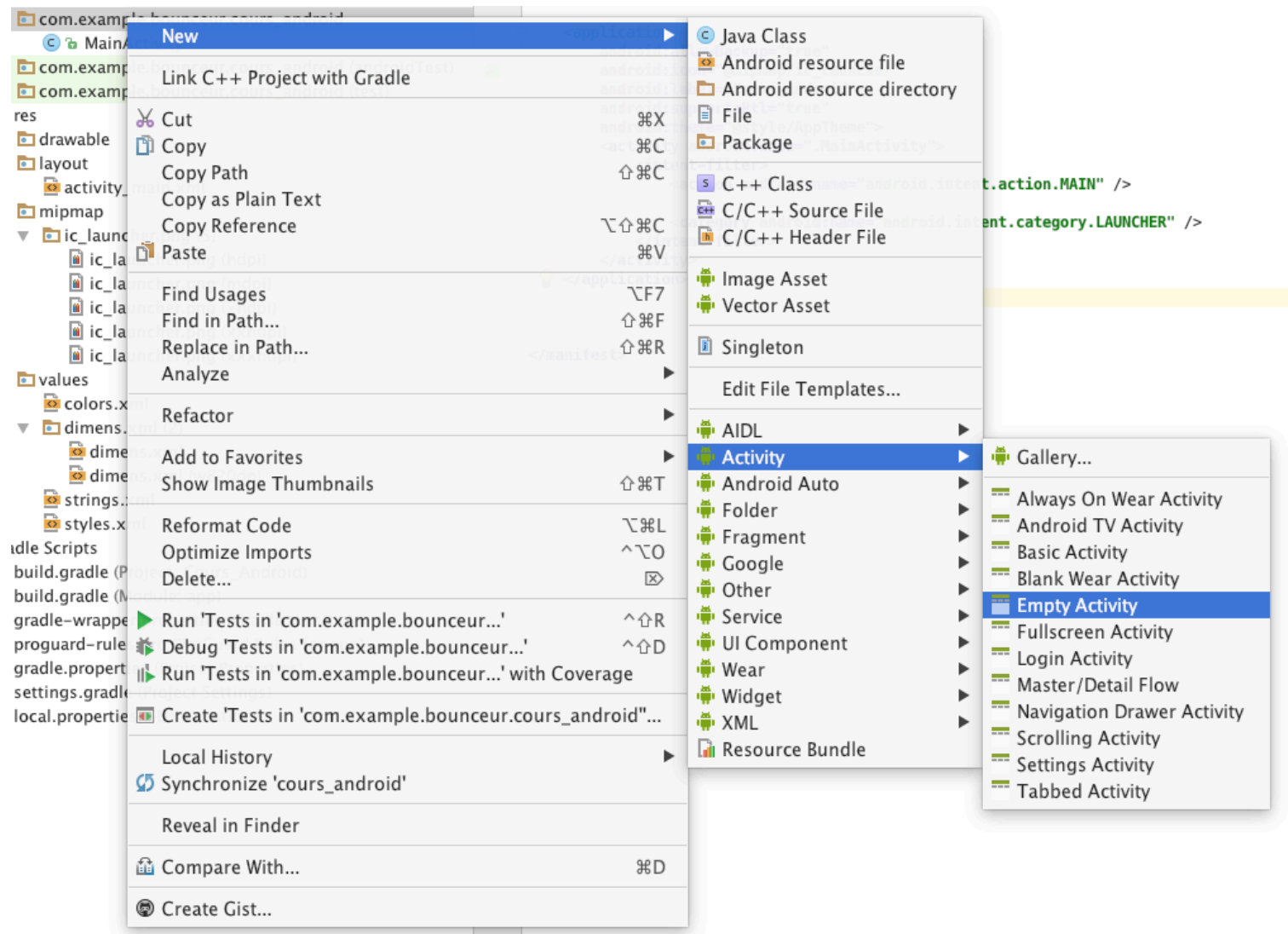
```
import android.app.Activity;
import android.os.Bundle;

public class ActiviteSimple extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getSupportActionBar().hide();
        getWindow().setFlags(
            WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN
        );
    }
}
```



# CRÉER UNE ACTIVITÉ


## ○ Méthode 1 :



# CRÉER UNE ACTIVITÉ

## ○ Méthode 1 :

New Android Activity

 **Configure Activity**  
Android Studio

Creates a new empty activity

Activity Name:

☒ Generate Layout File

Layout Name:

☐ Launcher Activity

☒ Backwards Compatibility (AppCompat)

Package name:

The name of the activity class to create

# CRÉER UNE ACTIVITÉ

## ○ Méthode 2 :

- Créer une classe Java classique
- Cette classe doit hériter de la classe Activity
- Ajouter la méthode onCreate(Bundle savedInstanceState)
- Au début de cette méthode, appeler la méthode onCreate de la classe mère
- Ajouter l'interface graphique

```
public class MonActivite extends Activity {  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```



# CRÉER UNE ACTIVITÉ

## ○ Méthode 2 :

- Une activité doit être déclarée dans le fichier de configuration : **AndroidManifest.xml**
- Un Intent-Filter doit être ajouté afin de pouvoir ajouter
  - Une Action (principale ou secondaire)
  - Une Catégorie (se lance automatiquement ou pas)

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name=".Main2Activity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

# CRÉER UNE ACTIVITÉ

## ○ Méthode 2 :

- Une activité doit être déclarée dans le fichier de configuration : **AndroidManifest.xml**
- Un Intent-Filter doit être ajouté afin de pouvoir ajouter
  - Une Action (principale ou secondaire)
  - Une Catégorie (se lance automatiquement ou pas)

```
<activity android:name=".MainActivity">  
  <intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
    <category android:name="android.intent.category.LAUNCHER" />  
  </intent-filter>  
</activity>
```

```
<activity android:name=".Main2Activity"></activity>
```

# ACTIVITÉ : AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bounceur.cours_android">

    <application

        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name=".MainActivity"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

# ACTIVITÉ : AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bounceur.cours_android">

    <application

        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name=".MainActivity"
            android:process="nom"
        >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

# ACTIVITÉ : AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bounceur.cours_android">

    <application

        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name=".MainActivity"
            android:process="nom"
            android:permission="android.permission.WAKE_LOCK"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

# ACTIVITÉ : AndroidManifest.xml

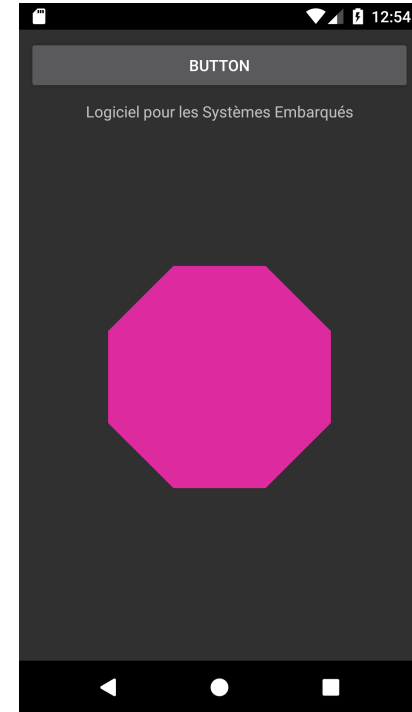
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fr.univ_brest.bounceur.projet_cours">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name=".MainActivity"
            android:theme="@style/Theme.Design.NoActionBar"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```



# ACTIVITÉ : AndroidManifest.xml

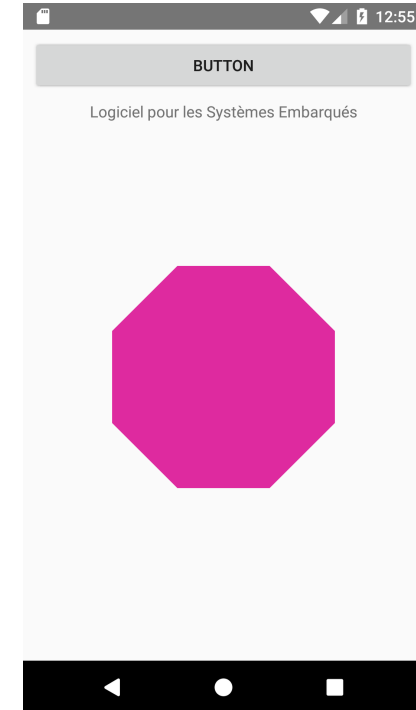
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fr.univ_brest.bounceur.projet_cours">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name=".MainActivity"
            android:theme="@style/Theme.Design.Light.NoActionBar"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```



# ACTIVITÉ : AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fr.univ_brest.bounceur.projet_cours">
```

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
```

```
        <activity
            android:name=".MainActivity"
            android:screenOrientation="portrait"
        >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

*portrait*  
*landscape*  
*...*

```
    </application>
```

```
</manifest>
```





# ACTIVITÉ : AndroidManifest.xml

- Priorité d'une activité :

```
<activity                                -1000 à 1000
    android:name=".MainActivity"
    android:label="Mon Activité">

    <intent-filter android:priority="999">

        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />

    </intent-filter>

</application>
```

# ACTIVITÉ : AndroidManifest.xml

- Label et Mode d'exécution :

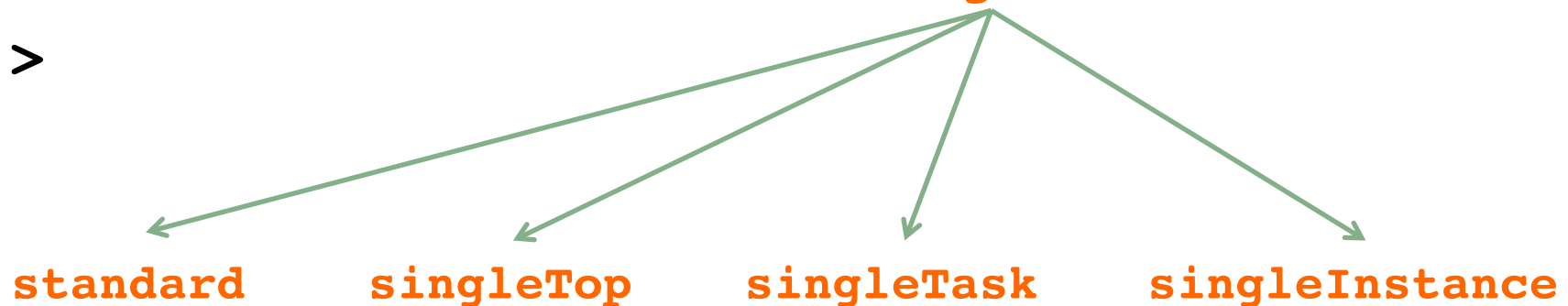
```
<activity
```

```
    android:name=".MainActivity"
```

```
    android:label="Mon Activité"
```

```
    android:launchMode="singleTask"
```

```
>
```

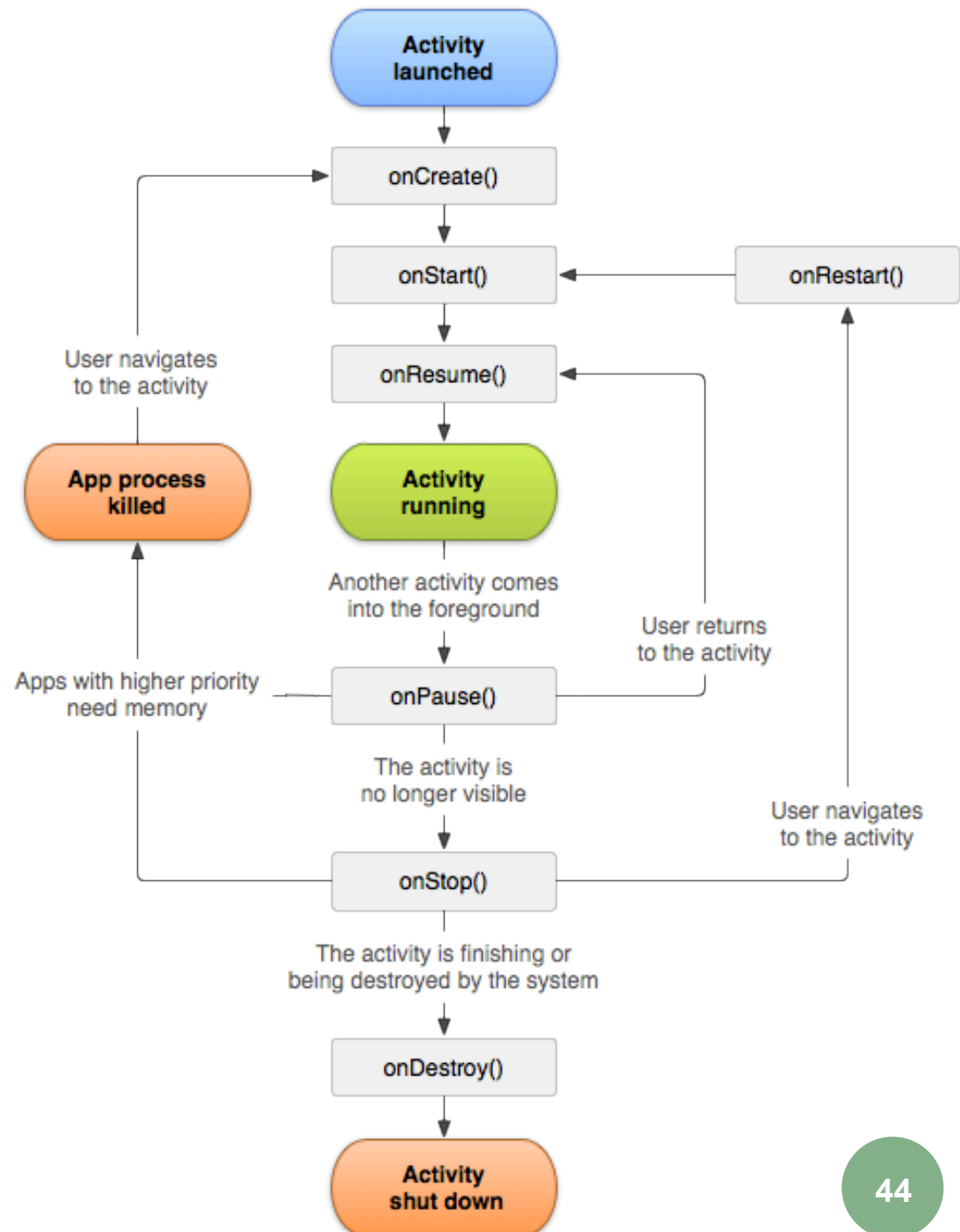


# CYCLE DE VIE D'UNE ACTIVITÉ

- Les états principaux d'une activité sont les suivants :
  - **active (active)** : activité visible qui détient le focus utilisateur et attend les entrées utilisateur. C'est l'appel à la méthode **onResume**, à la création ou à la reprise après pause qui permet à l'activité d'être dans cet état. Elle est ensuite mise en pause quand une autre activité devient active grâce à la méthode **onPause** ;
  - **suspendue (paused)** : activité au moins en partie visible à l'écran mais qui ne détient pas le focus. La méthode **onPause** est invoquée pour entrer dans cet état et les méthodes **onResume** ou **onStop** permettent d'en sortir ;
  - **arrêtée (stopped)** : activité non visible. C'est la méthode **onStop** qui conduit à cet état.

# CYCLE DE VIE D'UNE ACTIVITÉ

- Le cycle de vie d'une activité est parsemé d'appels aux méthodes relatives à chaque étape de sa vie. Il informe ainsi le développeur sur la suite des événements et le travail qu'il doit accomplir.



## Activité : squelette (1/9)

```
public final class MonActivite extends Activity {  
    /**  
     * Appelée lorsque l'activité est créée.  
     * Permet de restaurer l'état de l'interface  
     * utilisateur grâce au paramètre savedInstanceState.  
     */  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // Placez votre code ici  
    }  
    ...  
}
```

## Activité : squelette (2/9)

```
public final class MonActivite extends Activity {  
    ...  
    /**  
     * Appelée lorsque que l'activité a fini son cycle de vie.  
     * C'est ici que nous placerons notre code de libération de  
     * mémoire, fermeture de fichiers et autres opérations  
     * de "nettoyage".  
     */  
    public void onDestroy() {  
        // Placez votre code ici  
        super.onDestroy();  
    }  
    ...  
}
```

## Activité : squelette (3/9)

```
public final class MonActivite extends Activity {  
    ...  
    /**  
     * Appelée lorsque l'activité démarre.  
     * Permet d'initialiser les contrôles.  
     */  
    public void onStart() {  
        super.onStart();  
        // Placez votre code ici  
    }  
    ...  
}
```

## Activité : squelette (4/9)

```
public final class MonActivite extends Activity {  
    ...  
    /**  
     * Appelée lorsque l'activité passe en arrière plan.  
     * Libérez les écouteurs, arrêtez les threads, votre activité  
     * peut disparaître de la mémoire.  
     */  
    public void onStop() {  
        // Placez votre code ici  
        super.onStop();  
    }  
    ...  
}
```



## Activité : squelette (5/9)

```
public final class MonActivite extends Activity {  
    ...  
    /**  
     * Appelée lorsque l'activité sort de son état de veille.  
     */  
    public void onRestart() {  
        super.onRestart();  
        // Placez votre code ici  
    }  
    ...  
}
```

## Activité : squelette (6/9)

```
public final class MonActivite extends Activity {  
    ...  
    /**  
     * Appelée lorsque que l'activité est suspendue.  
     * Stoppez les actions qui consomment des ressources.  
     * L'activité va passer en arrière-plan.  
     */  
    public void onPause() {  
        // Placez votre code ici  
        super.onPause();  
    }  
    ...  
}
```

## Activité : squelette (7/9)

```
public final class MonActivite extends Activity {  
    ...  
    /**  
     * Appelée après le démarrage ou une pause.  
     * Relancez les opérations arrêtées (threads).  
     * Mettez à jour votre application et vérifiez vos écouteurs.  
     */  
    public void onResume() {  
        super.onResume();  
        // Placez votre code ici  
    }  
    ...  
}
```

The left side of the page features a series of vertical bars of varying heights and shades of green. To the right of these bars are several green circles of different sizes. One circle contains the number 52.

**POUR FINIR**

52

# AFFICHER UNE BOITE DE DIALOGUE

```
AlertDialog builder = new AlertDialog.Builder(this).create();  
builder.setTitle("Information");  
builder.setMessage("Value of x is "+x);  
builder.setIcon(R.drawable.triangle);  
builder.show();
```



# AFFICHER UNE BOITE DE DIALOGUE

Voulez-vous quitter ?

ANNULER

NON

OUI

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setMessage("Voulez-vous quitter ?");

builder.setPositiveButton("Oui", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // Action pour oui
    }
});
builder.setNegativeButton("Non", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // Action pour non
    }
});
builder.setNeutralButton("Annuler", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // Action pour annuler
    }
});

builder.show();
```

# AFFICHER UN MESSAGE

- Par un Toast

```
Toast.makeText(this, "Salut !", Toast.LENGTH_SHORT).show();
```

```
Toast.makeText(this, "Salut !", Toast.LENGTH_LONG).show();
```

# AFFICHER UN MESSAGE

- Par un SnackBar

```
Snackbar snackbar = Snackbar.make(findViewById(R.id.lay),  
"Welcome !", Snackbar.LENGTH_LONG);
```

```
snackbar.show();
```



# AFFICHER UN MESSAGE

## ○ Par un SnackBar + Action

```
View parentLayout = findViewById(R.id.lay);

Snackbar.make(parentLayout, "This is main activity", Snackbar.LENGTH_LONG)
    .setAction("CLOSE", new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //...
        }
    })
    .setActionTextColor(Colo.RED)
    .show();
```

# AFFICHER UNE NOTIFICATION

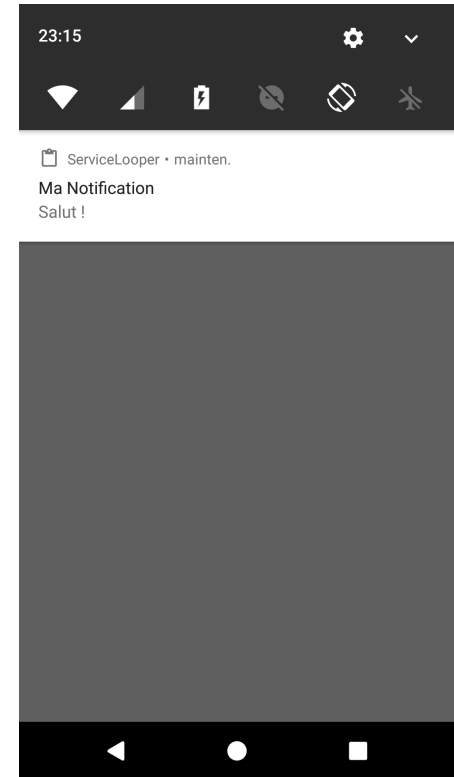
```
NotificationCompat.Builder mBuilder = new  
NotificationCompat.Builder(this);
```



```
mBuilder.setSmallIcon(R.drawable.notif_icone);  
mBuilder.setContentTitle("Ma notification");  
mBuilder.setContentText("Salut!");  
mBuilder.setTicker("Ticker");
```

```
NotificationManager mNotificationManager = (NotificationManager)  
getSystemService(this.NOTIFICATION_SERVICE);
```

```
mNotificationManager.notify(1, mBuilder.build());
```



# AFFICHER UNE NOTIFICATION

```
NotificationCompat.Builder mBuilder = new  
NotificationCompat.Builder(this);
```

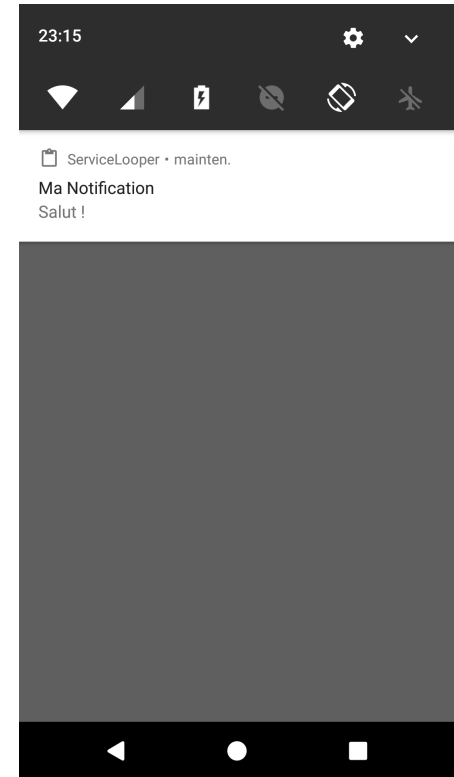


```
mBuilder.setSmallIcon(R.drawable.notif_icone);  
mBuilder.setContentTitle("Ma notification");  
mBuilder.setContentText("Salut!");  
mBuilder.setTicker("Ticker");
```

```
Notification notification = mBuilder.build();
```

```
NotificationManager mNotificationManager = (NotificationManager)  
getSystemService(this.NOTIFICATION_SERVICE);
```

```
mNotificationManager.notify(1, notification);
```



# AFFICHER UNE NOTIFICATION

```
NotificationCompat.Builder mBuilder = new  
NotificationCompat.Builder(this);
```

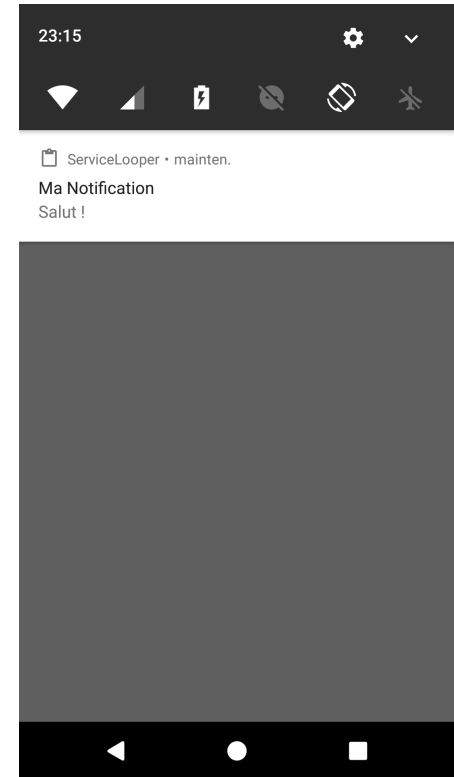


```
mBuilder.setSmallIcon(R.drawable.notif_icone)  
.setContentTitle("Ma notification")  
.setContentText("Salut!")  
.setTicker("Ticker");
```

```
Notification notification = mBuilder.build();
```

```
NotificationManager mNotificationManager = (NotificationManager)  
getSystemService(this.NOTIFICATION_SERVICE);
```

```
mNotificationManager.notify(1, notification);
```

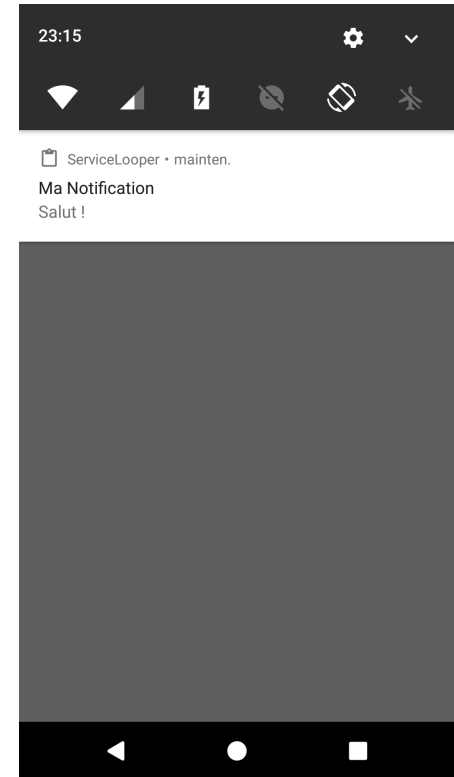


# AFFICHER UNE NOTIFICATION

```
Notification notification =  
new NotificationCompat.Builder(this)  
    .setSmallIcon(R.drawable.notif_icone)  
    .setContentTitle("Ma notification")  
    .setContentText("Salut!")  
    .setTicker("Ticker")  
    .build();
```

```
NotificationManager mNotificationManager = (NotificationManager)  
getSystemService(this.NOTIFICATION_SERVICE);
```

```
mNotificationManager.notify(1, notification);
```

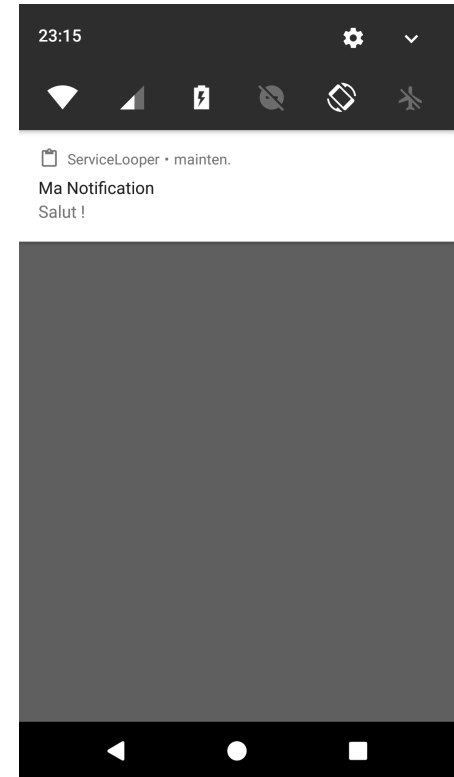


# AFFICHER UNE NOTIFICATION

```
Notification notification =  
new NotificationCompat.Builder(this)  
    .setSmallIcon(R.drawable.notif_icone)  
    .setContentTitle("Ma notification")  
    .setContentText("Salut!")  
    .setTicker("Ticker")  
    .setOngoing(true)  
    .build();
```

```
NotificationManager mNotificationManager = (NotificationManager)  
getSystemService(this.NOTIFICATION_SERVICE);
```

```
mNotificationManager.notify(1, notification);
```

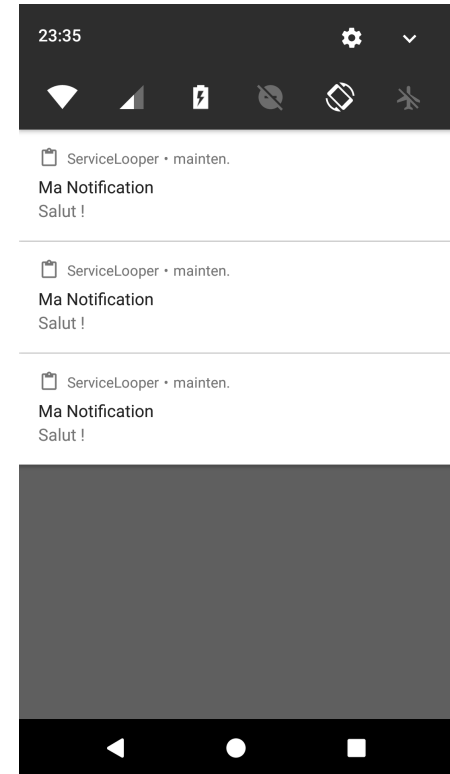


# AFFICHER PLUSIEURS NOTIFICATIONS

```
Notification notification =  
new NotificationCompat.Builder(this)  
    .setSmallIcon(R.drawable.notif_icone)  
    .setContentTitle("Ma notification")  
    .setContentText("Salut!")  
    .setTicker("Ticker")  
    .setOngoing(true)  
    .build();
```

```
NotificationManager mNotificationManager = (NotificationManager)  
getSystemService(this.NOTIFICATION_SERVICE);
```

```
mNotificationManager.notify(1, notification);  
mNotificationManager.notify(2, notification);  
mNotificationManager.notify(3, notification);
```



# AFFICHER UNE NOTIFICATION (AVEC SON)

```
Uri alarmSound =
    RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);

Notification notification =
    new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.notif_icone)
        .setContentTitle("Ma notification")
        .setContentText("Salut!")
        .setTicker("Ticker")
        .setOngoing(true)
        .setAutoCancel(true)
        .setLights(Color.BLUE, 500, 500)
        .setStyle(new NotificationCompat.InboxStyle())
        .setSound(alarmSound);
    .build();

NotificationManager mNotificationManager = (NotificationManager)
    getSystemService(this.NOTIFICATION_SERVICE);

mNotificationManager.notify(1, notification);
mNotificationManager.notify(2, notification);
mNotificationManager.notify(3, notification);
```



## AFFICHER UNE NOTIFICATION

- On peut lier une notification à une activité
- La notification peut interagir avec l'activité
- Exemple : mini lecteur de musique

- On utilise un `PendingIntent`
- Et la propriété `setContentIntent`

→ Cette partie sera traitée dans le cours sur les Services

## FORCER L'ACTIVITÉ À RESTER ALLUMÉE (BLOQUER LE MODE VEILLE)

- Il suffit juste d'ajouter un flag :

```
getWindow().addFlags(  
    WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON  
);
```

- On peut le faire aussi à partir d'un layout :

```
<android.support.constraint.ConstraintLayout xmlns:android="http..."  
    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:keepScreenOn="true"  
    tools:context="com.bounceur.servicelooper.MainActivity"  
>
```

