

## 2. États d'une application

---

Présentation des états par lesquels transite une activité, incluant leur sauvegarde et restauration.

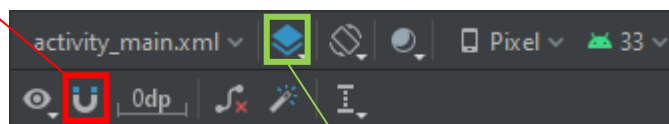
*Android Studio Hedgehog Essentials, Kotlin Edition* : chapitres 9, 10, 19 à 22

### 2.1. Présentation *Powerpoint*

1. Présentez la présentation portant sur l'architecture du OS ainsi que le cycle de vie d'une application *Android*

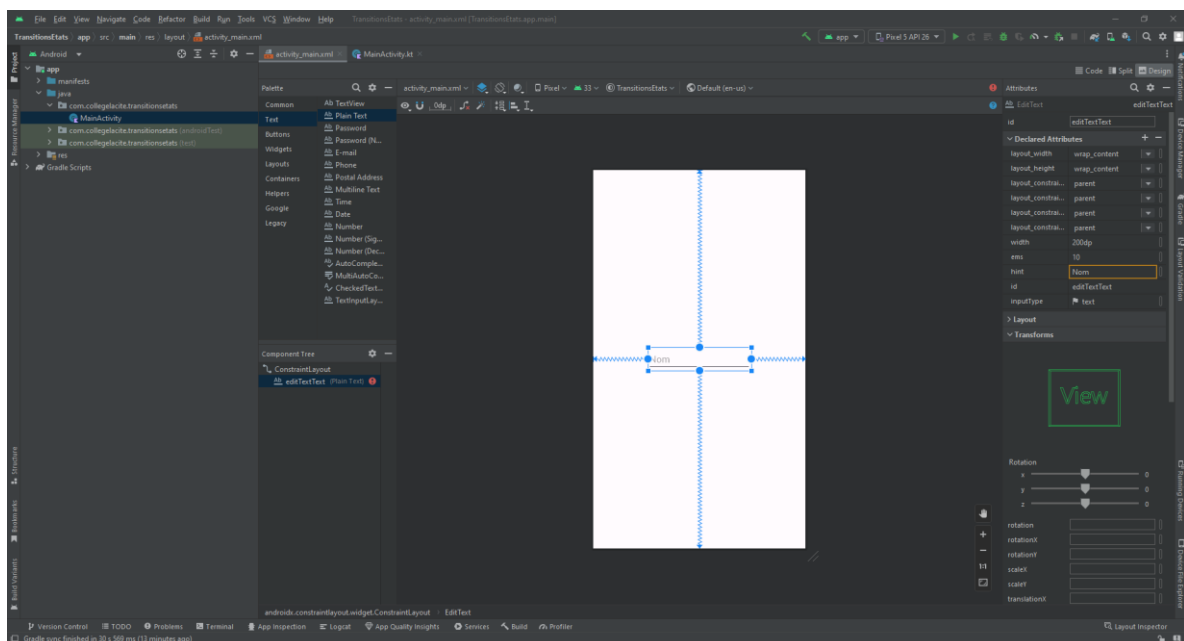
### 2.2. Créer une nouvelle application

2. Indiquez aux étudiants qu'ils doivent effectuer les mêmes opérations que l'instructeur et qu'ils s'assurent que leur application résultante est fonctionnelle
3. Démarrez *Android Studio* et sélectionnez « New Project ». Si un projet antérieur est déjà chargé au démarrage de l'application, fermez celui-ci au préalable via File » Close Project
4. Dans la fenêtre de gabarits, expliquez quelques types de projets d'applications *Android*. Sélectionnez « Empty Views Activity ». Appuyez ensuite le bouton « Next »
5. Nommez le projet « *TransitionsEtats* », entrez le nom de domaine du collège et sélectionnez un répertoire où créer le projet, sélectionnez Kotlin comme langage, et sélectionnez « API 26: Android 8.0 (Oreo) » comme SDK minimal (ne pas activer les autres options). Appuyez ensuite le bouton « Finish »
6. Assurez-vous que le layout `activity_main.xml` est chargé dans l'éditeur visuel, et que l'*Autoconnection* de l'éditeur *Design* est actif



- a. Vous pouvez aussi cacher l'affichage « Blueprint » de l'activité dans l'éditeur *Design*

7. Supprimez le widget « *Hello World!* » du contenu actuel, et déposez un Plain Text au centre de l'activité (à l'aide des lignes guides)
  - a. Expliquez que les widgets de type `EditText` ont un attribut `inputType` qui permet de contrôler les caractères acceptés par le composant ; sélectionnez la valeur `text` à cette propriété via le panneau de propriétés du composant
  - b. Via le panneau de propriétés, attribuez la valeur `200dp` à l'attribut `width` du widget
  - c. Effacez le contenu de l'attribut `text` du widget
  - d. Entrez la chaîne « Nom » dans l'attribut `hint` du widget



8. Affichez le code source de l'activité dans le fichier `MainActivity.kt`. Ce fichier devrait déjà être chargé dans l'éditeur textuel (via un onglet au-dessus de l'éditeur). Sinon, double-cliquez sur ce fichier dans le *Project Panel* (app » java » `com.collegelacite.transitionsetats » MainActivity`)
  - a. Soulignez que la classe `MainActivity` est dérivée de la classe *Android AppCompatActivity*
  - b. Soulignez aussi que la classe `MainActivity` ne surcharge qu'une seule fonction membre de cycle de vie, `onCreate()`

9. Modifiez `onCreate()` (ainsi que les autres fonctions liées au cycle de vie) afin qu'elle affiche un message de journalisation dans le panneau *LogCat* de *Android Studio* :

```
package com.collegelacite.transitionsetats

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log

class MainActivity : AppCompatActivity() {
    private val TAG = "TransitionsEtatsTAG"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        Log.i(TAG, "onCreate a été invoquée");
    }

    override fun onStart() {
        super.onStart()
        Log.i(TAG, "onStart a été invoquée")
    }

    override fun onResume() {
        super.onResume()
        Log.i(TAG, "onResume a été invoquée")
    }

    override fun onPause() {
        super.onPause()
        Log.i(TAG, "onPause a été invoquée")
    }

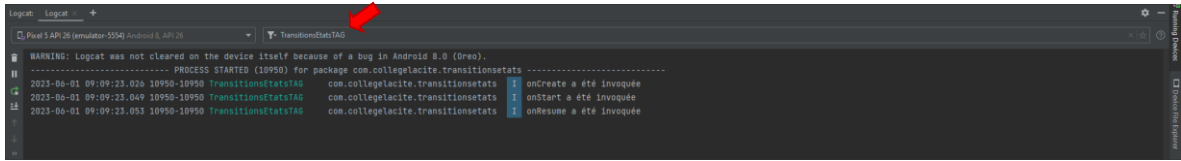
    override fun onStop() {
        super.onStop()
        Log.i(TAG, "onStop a été invoquée")
    }


    override fun onRestart() {
        super.onRestart()
        Log.i(TAG, "onRestart a été invoquée")
    }

    override fun onDestroy() {
        super.onDestroy()
        Log.i(TAG, "onDestroy a été invoquée")
    }
}
```

10. Exécutez l'application, affichez le *Logcat* (via l'onglet au bas de l'environnement de développement), puis identifiez les entrées de journaux liés au cycle de vie
- Si le `EditText` n'est pas centré dans l'activité, c'est que vous avez oublié d'activer *Autoconnect* de l'éditeur *Design*.

11. Pour n'affichez que les journaux liés au cycle de vie, appliquez un filtre d'affichage correspondant à la variable `TAG` de notre code dans le champ *Regex* du moniteur



12. Dans le *Running Devices*, pressez le bouton *Home* (  ) puis soulignez les nouvelles entrées de journaux dans *Logcat*

- Réaffichez l'application pour montrer les changements d'états occasionnés
- Appliquez une rotation de l'appareil pour montrer que l'activité est supprimée puis recrée (afin qu'elle s'adapte au changement de configuration de l'appareil)

13. Expliquez que le contenu du `EditText` est restauré parce que le widget dispose de l'attribut **saveEnabled** qui est `true` par défaut

- Changez la valeur de cet attribut à `false` puis redémarrez l'application pour démontrer que le contenu du widget n'est plus conservé lors des rotations
- Expliquez que les widgets ne disposent pas tous d'un attribut `saveEnabled`. Ceux n'ayant pas cet attribut ne peuvent pas d'eux-mêmes conserver leur état. Dans ces cas il faut le faire par programmation

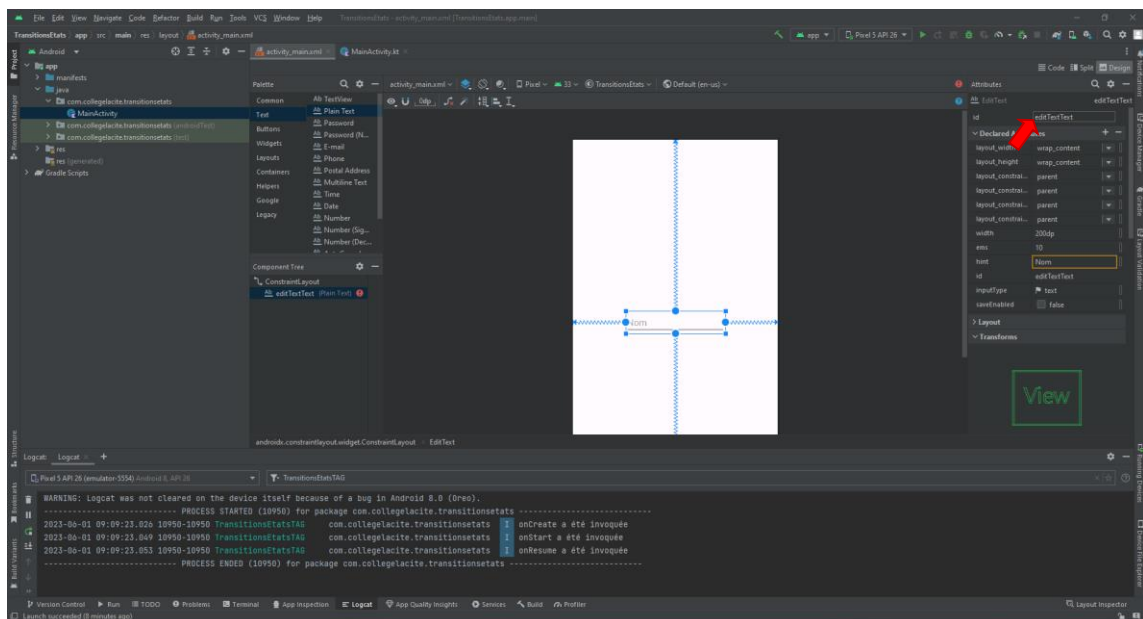
## 2.3. Sauvegarde et restauration d'état

14. Ajoutez les deux surcharges suivantes au code source de l'activité

```
override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    Log.i(TAG, "onSaveInstanceState a été invoquée")
}

override fun onRestoreInstanceState(savedInstanceState: Bundle) {
    super.onRestoreInstanceState(savedInstanceState)
    Log.i(TAG, "onRestoreInstanceState a été invoquée")
}
```

15. Exécutez à nouveau l'application puis effectuer une rotation d'appareil pour démontrer que l'état de l'activité est sauvegardé avant sa suppression, puis restaurée à sa création
  - a. Expliquez ce qu'est le `Bundle` d'une application (emphase sur le stockage de paires `<clé:valeur>` où la clé est une chaîne et la valeur un type implémentant `Parcelable`)
  - b. Expliquez que tous les widgets d'*Android* supportent automatiquement la restauration d'état en autant que le widget ait un `id` et l'activité qui les contient surcharge au minimum les deux fonctions membres `onSaveInstanceState()` et `onRestoreInstanceState()`
16. Expliquez que *Android Studio* attribue un identificateur unique à chaque widget déposé dans une activité ; par défaut, c'est le type du widget suivi d'un compteur au besoin :



- a. Montrez qu'il est possible de renommer un widget via sa propriété `id`, mais conservez `editTextText` comme `id`
17. Expliquez que toutes les ressources constituant une application sont compilées dans une classe nommée `R`, incluant celles définissant le *layout* de l'activité. On peut ainsi obtenir une référence à un widget de l'activité via la classe `R`

18. Modifiez les surcharges de sauvegarde et de restauration de l'activité afin de conserver l'état du EditText :

```
package com.collegelacite.transitionsetats

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.EditText

class MainActivity : AppCompatActivity() {

    ...

    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        Log.i(TAG, "onSaveInstanceState a été invoquée")

        val textBox: EditText = findViewById(R.id.editTextText)
        val userText = textBox.text
        outState.putCharSequence("savedText", userText)
    }

    override fun onRestoreInstanceState(savedInstanceState: Bundle) {
        super.onRestoreInstanceState(savedInstanceState)
        Log.i(TAG, "onRestoreInstanceState a été invoquée")

        val textBox: EditText = findViewById(R.id.editTextText)
        val userText = savedInstanceState.getCharSequence("savedText")
        textBox.setText(userText)
    }
}
```

19. Exécutez l'application pour démontrer que l'état du widget est effectivement conservé à la suite d'une rotation de l'appareil

**Évaluation formative 02** Indiquez aux étudiants qu'ils peuvent récupérer l'énoncé de l'exercice (en format PDF) sur le portail éducatif du collège. Ils devraient pouvoir solutionner l'exercice sans aide en se référant aux chapitres 19 à 22 du livre de référence