ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

TP2

Création d'une première application Android : Cycle de vie d'une activité

Objectif

Découvrir l'environnement Android Studio en créant une première application simple.
Observer et comprendre le **cycle de vie d'une activité** à travers les logs d'exécution.

Compétences visées

- Créer un projet Android sous Android Studio
- Comprendre la structure d'un projet Android (Code, Ressources, Manifest)
- Identifier les principales méthodes du cycle de vie (`onCreate()`, `onStart()`, `onResume()`, … etc.)
- Utiliser le **Logcat** pour suivre les messages système et développeur

---

Matériel et outils nécessaires *(résultats du TP1)*

- Android Studio (version récente recommandée)
- SDK Android installé (API 35 ou supérieure)
- Un AVD (émulateur Android) ou un smartphone configuré en mode développeur

---

Étapes à suivre

Étape 1 – Création du projet

1. Ouvrir **Android Studio → New Project**
2. Choisir le modèle **Empty Views Activity**
3. Renseigner les informations suivantes :
    - Nom du projet : `CycleDeVieActivity`
    - Langage : **Java**
    - API minimale : **API 24 (Android 7.0 Nougat)**

4. Cliquer sur **Finish** pour générer le projet.

Étape 2 – Explorer la structure du projet

Identifier les principaux éléments générés :

- `MainActivity.java` → code Java principal
- `activity_main.xml` → interface utilisateur
- `AndroidManifest.xml` → métadonnées de l'application

Étape 3 – Modifier la classe MainActivity

Ajouter les méthodes du cycle de vie et des logs dans chacune.

Code Java (génération automatique du code avec les outils Android Studio)

```
public class MainActivity extends AppCompatActivity {

    private static final String TAG = "CycleDeVie";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(TAG, "onCreate() appelé");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d(TAG, "onStart() appelé");
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.d(TAG, "onResume() appelé");
    }
    @Override
    protected void onPause() {
        super.onPause();
        Log.d(TAG, "onPause() appelé");
    }
    @Override
    protected void onStop() {
        super.onStop();
        Log.d(TAG, "onStop() appelé");
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "onDestroy() appelé");
    }
```

```
    @Override
    protected void onRestart() {
        super.onDestroy();
        Log.d(TAG, "onRestart() appelé");
    }
}
```

Étape 4 – Exécution et observation

1. Lancer l'application (**Run** ▸)
2. Observer les messages dans l'onglet **Logcat**
    - Filtrer par `CycleDeVie` pour afficher uniquement vos logs
3. Noter les différentes méthodes appelées selon les actions suivantes :
    - Démarrage de l'activity
    - Passage à une autre activity
    - Retour à activity
    - Fermeture de l'activity

Étape 5 – Analyse et conclusion

- Expliquer l'ordre d'exécution des méthodes observées.
- Identifier les moments clés :
    - Initialisation : `onCreate()`
    - Affichage : `onRestart()` / `onResume()`
    - Arrêt : `onPause()` / `onStop()`
    - Destruction : `onDestroy()`
- Bonus : Implémentez les autres méthodes de sauvegarde automatique de l'état.
- Répondre : *Pourquoi Android gère-t-il ainsi le cycle de vie d'une activité ?*

Livrables attendus

- Code source complet de `MainActivity.java`
- Capture d'écran du Logcat montrant l'ordre des appels

- Tableau récapitulatif du cycle de vie observé
- Brève conclusion (5 à 10 lignes) sur le comportement constaté

Durée estimée

Environ **1h30 à 2h**