Tutorial du création de la première application mobile sous Android :

I. Etapes de création :

Pour créer votre première application mobile Android sous Android Studio, vous pouvez suivre ces étapes simples :

1. **Installer Android Studio** : Téléchargez et installez Android Studio à partir du site officiel : Android Studio Download.
2. **Créer un nouveau projet** :
    - Lancez Android Studio.
    - Cliquez sur "Start a new Android Studio project" ou sélectionnez "File" > "New" > "New Project".
    - Choisissez "Empty Activity" comme modèle de projet.
    - Cliquez sur "Next", puis entrez les détails de votre application (nom, package name, etc.).
    - Cliquez sur "Finish" pour créer votre projet.
3. **Configurer l'émulateur ou un appareil réel** :
    - Configurez un émulateur en cliquant sur "Tools" > "AVD Manager" et en créant un nouvel appareil virtuel Android, ou connectez un appareil Android réel à votre ordinateur en activant le mode développeur et le débogage USB.
4. **Éditer l'interface utilisateur** :
    - Ouvrez le fichier XML de l'activité principale (**activity_main.xml**) dans le dossier **res/layout**.
    - Utilisez l'éditeur de mise en page pour ajouter des composants d'interface utilisateur tels que des boutons, des textes, des images, etc.
5. **Ajouter du code Java** :
    - Ouvrez le fichier Java de l'activité principale (**MainActivity.java**) dans le dossier **java/<votre_package_name>**.
    - Ajoutez le code Java pour gérer les interactions utilisateur, les calculs, etc.
6. **Exécuter l'application** :
    - Cliquez sur le bouton "Run" (icône de triangle vert) dans Android Studio.
    - Sélectionnez votre appareil virtuel ou appareil réel et cliquez sur "OK".
    - Attendez que l'application soit installée et exécutée sur l'appareil.
7. **Tester et déboguer** :

- Utilisez l'émulateur ou l'appareil réel pour tester votre application.
- Utilisez les outils de débogage d'Android Studio pour résoudre les problèmes éventuels.

8. **Déployer votre application** :
    - Une fois votre application prête, générez un fichier APK en sélectionnant "Build" > "Build Bundle(s) / APK(s)" > "Build APK(s)".
    - Vous pouvez ensuite distribuer cet APK sur le Google Play Store ou sur d'autres canaux de distribution.

II. Utilité des fichiers

Dans un projet Android créé avec Android Studio, différents fichiers ont des rôles spécifiques. Voici une explication de l'utilité de chaque fichier couramment trouvé dans un projet Android :

1. **AndroidManifest.xml** : Ce fichier décrit les caractéristiques de votre application et de chacune de ses composantes. Il contient des informations telles que le nom du package, les permissions requises, les composants de l'application (activités, services, récepteurs de diffusion) et les configurations de sécurité.

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application_java"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication_java"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

2. **build.gradle (Project)** : Ce fichier définit la configuration du projet, y compris les versions des plugins Gradle utilisés pour la compilation de l'application. Il peut également inclure la configuration des référentiels de dépendances externes.
3. **build.gradle (Module)** : Ce fichier définit la configuration spécifique au module de l'application, y compris les dépendances nécessaires à la compilation de l'application, les versions du SDK Android cible, les paramètres de compilation et d'autres configurations spécifiques au module.
4. **MainActivity.java** : Ce fichier contient la classe Java principale de votre application, généralement une activité. Il est responsable de l'initialisation de l'interface utilisateur de l'application et de la gestion des interactions utilisateur.

```java
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

    }
    1 usage
    public void bugetAge(View view) {
        EditText editAge =(EditText)findViewById(R.id.ageText_id);
        int year= Integer.parseInt(editAge.getText().toString());
        int age=2024-year;
        Toast.makeText( context: this, text: "age"+String.valueOf(age),Toast.LENGTH_LONG).show();

    }
}
```

5. **activity_main.xml** : Ce fichier est le fichier de mise en page de l'activité principale de votre application. Il définit l'apparence de l'interface utilisateur à l'aide de balises XML, définissant les vues telles que les boutons, les textes, les images, etc.

```xml
<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:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">
    <EditText
        android:id="@+id/ageText_id"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"

        android:text="Enter your year"
        android:textColorHint="@color/design_default_color_error" />

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="bugetAge"
        android:text="Get Age " />
```

6. **res/layout/** : Ce répertoire contient tous les fichiers XML de mise en page de votre application. Chaque fichier correspond généralement à une activité ou à un composant de l'interface utilisateur.
7. **res/values/strings.xml** : Ce fichier contient les chaînes de caractères utilisées dans votre application. Il est recommandé d'utiliser des ressources de chaînes pour faciliter la traduction et la gestion des chaînes.
8. **res/drawable/** : Ce répertoire contient les ressources graphiques utilisées dans votre application, telles que les icônes, les images et les fichiers XML de forme.
9. **res/menu/** : Ce répertoire contient les fichiers XML de menu de votre application, qui définissent les éléments de menu qui apparaissent dans la barre d'action de votre application.
10. **res/mipmap/** : Ce répertoire contient les icônes de l'application dans différentes densités de pixels. Ces icônes sont utilisées pour représenter l'application sur l'écran d'accueil de l'appareil.
11. **res/drawable-v24/** : Ce répertoire est similaire à res/drawable/, mais contient des ressources spécifiques à la version 24 (Android 7.0) et supérieure.