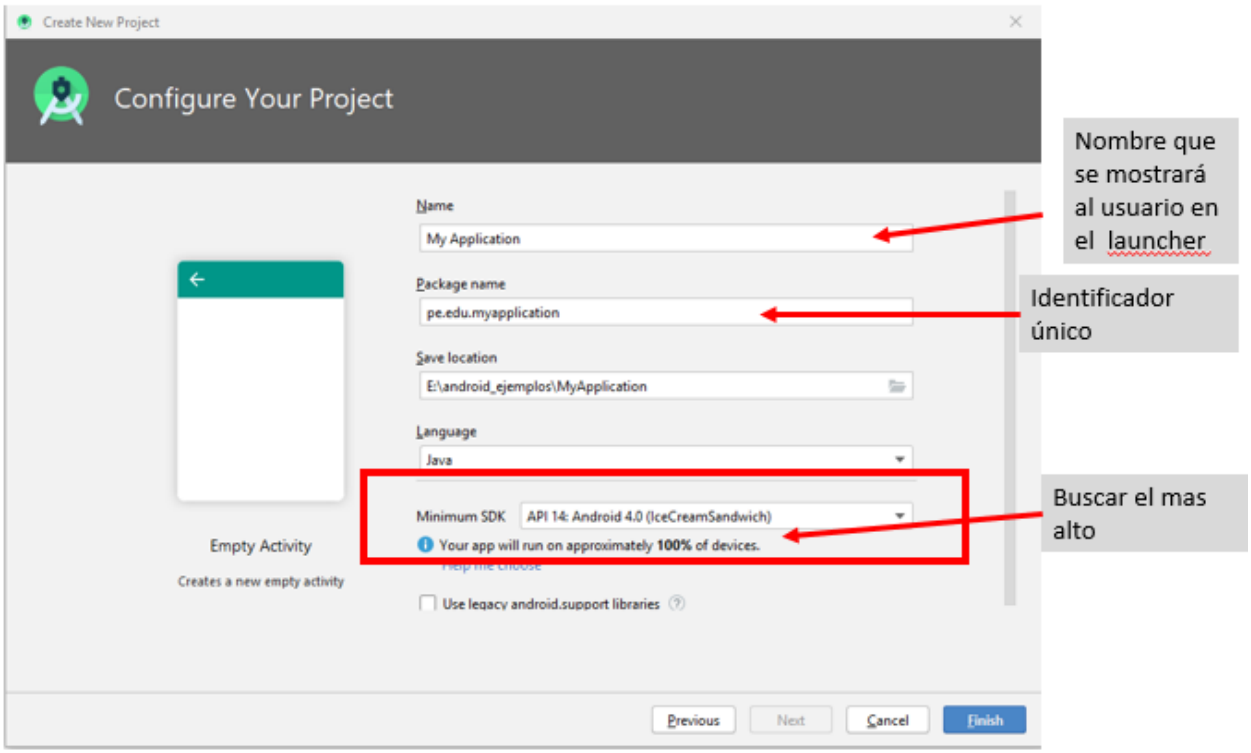
SEMANA 1 – PRACTICA 1

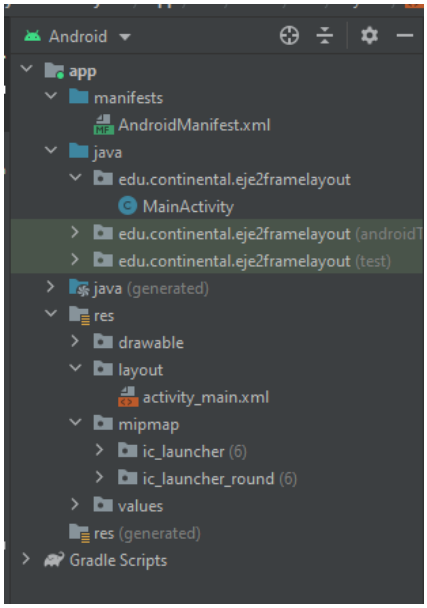
Instrucciones:

1. Instala Android Studio.

2. Crea un nuevo proyecto de Android Studio:
   - En el cuadro de diálogo completa como se muestra.



3. En el lado derecho tienes la vista del proyecto Android, el cual iremos revisando en sus componentes principales:



Apertura cada de las carpetas y archivos que se indican a continuación:

3.1 AndroidManifest.xml:

Enlazador de los archivos, los describe e indica como interactúan.

- Guarda configuración general de la aplicación.

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="testGame"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.TestGame">
        <activity
            android:name=".MainActivity"
            android:theme="@android:style/Theme.Holo.Light.NoActionBar.Fullscreen"
            android:exported="true"  >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

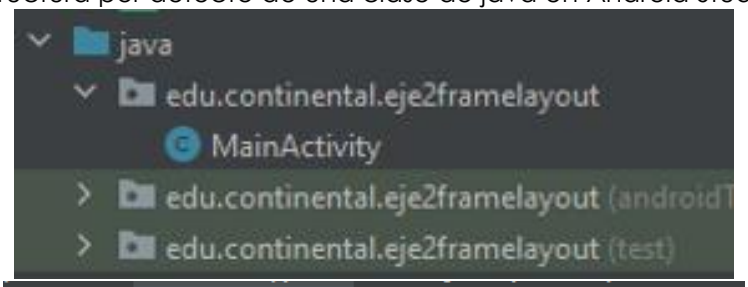
TAREA A EFECTUAR:

Cambia la línea **Android.label por : MiPrimeraApp**

3.2 Carpeta java

Guarda las clases java que definen la lógica de la app.
Agrupados por packages.
Si son muchas clases se recomienda crear packages.

Estructura por defecto de una clase de java en Android Studio.



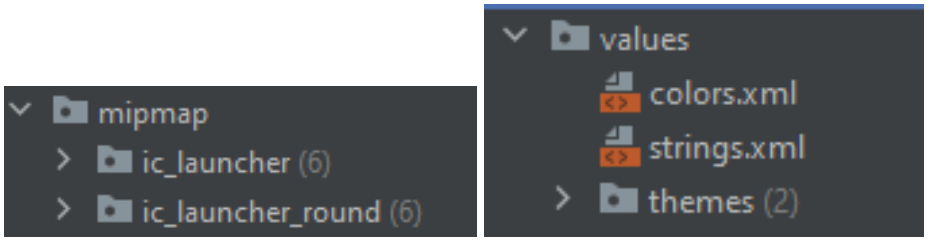

```java
package edu.continental.eje2framelayout;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

TAREA A EFECTUAR:

Crea un nuevo package llamado **Entidades**, para ello efectúa click derecho sobre la carpeta Java → new → Package, ingresa el nuevo nombre del package.

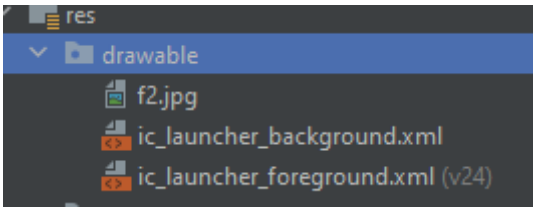
3.3 Res:

Todos los recursos de la aplicación se almacenan bajo la carpeta res/del proyecto. Dentro de esta carpeta encontramos diferentes subcarpetas para distintos tipos de recursos. Existen nueve tipos principales de recursos que tendrán su propia subcarpeta: valores simples, Drawables, layouts, animaciones, etilos, menús, searchables, XML y recursos raw. Al compilar nuestra aplicación estos recursos serán incluidos en el paquete apk que puede ser instalado en el dispositivo. Durante el proceso de compilación se generará también una clase R que contendrá referencias a cada uno de los recursos. Esto nos permitirá referenciar a los recursos desde nuestro código fuente



Drawable →
una subcarpeta por cada densidad de la pantalla, xml o mapa de bits.

- Mipmap → conserva, para el icono de la app.
- Values: definir textos, arrays, colores, estilos.



TAREA A EFECTUAR:

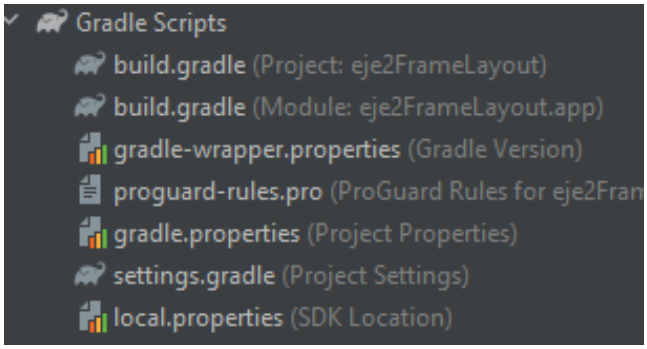
Crea un nuevo color en colors.xml

1. Ingresa a res → values → colors.xml
2. Ingresa el texto que se muestra en la línea 10.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="purple_200">#FFBB86FC</color>
4      <color name="purple_500">#FF6200EE</color>
5      <color name="purple_700">#FF3700B3</color>
6      <color name="teal_200">#FF03DAC5</color>
7      <color name="teal_700">#FF018786</color>
8      <color name="black">#FF000000</color>
9      <color name="white">#FFFFFFFF</color>
10     <color name="fondoFormulario">#2F3068</color>
11
12 </resources>
```

Gradle - scripts

El sistema de compilación de Android compila recursos y código fuente de la app y los empaqueta en APK que puedes probar, implementar, firmar y distribuir. Android Studio usa Gradle, un paquete de herramientas de compilación avanzadas, para automatizar y administrar el proceso de compilación.



Detalles a tener en cuenta Gradle Scripts

- **compileSdkVersion:** es la versión de Android que utilizan las herramientas de compilación (Gradle) para compilar la aplicación para su lanzamiento, ejecución o depuración.
- **applicationId**: es el identificador único de la aplicación cuando ésta sea publicada en Google Play. No pueden existir 2 aplicaciones publicadas con el mismo identificador.
- **minSdkVersion**: es la versión mínima del sistema operativo Android necesaria para ejecutar la aplicación. La aplicación no podrá ser instalada en un dispositivo que tenga una versión inferior a ésta.
- **targetSdkVersion:** conocida como versión SDK destino, es la versión de Android en la que se creó la aplicación para que se ejecute. Indica que hemos probado nuestra aplicación hasta la versión que especificamos en esta propiedad.
- **versionCode:** es un número que es usado para determinar si una versión es más reciente que otra. Este numero no es mostrado a los usuarios pero sirve para definir el numero de versión dentro de la Play Store.
- **versionName:** es una cadena de texto, su único propósito es mostrar el número de versión de la aplicación a los usuarios de Google Play.

targetSdkVersion

Para todos los propósitos prácticos, en la mayoría de las aplicaciones deberemos establecer targetSdkVersion a la última versión de la API. Esto asegurará que nuestra aplicación se vea lo mejor posible en los dispositivos Android más recientes. Si no especificamos targetSdkVersion , por defecto es minSdkVersion .

Debe por tanto cumplirse:

minSdkVersion <= targetSdkVersion <= compileSdkVersion

Lo ideal es:

minSdkVersion (la menor posible) <= targetSdkVersion == compileSdkVersion (la versión más reciente de SDK)

TAREA A EFECTUAR:

1. Ingresa a gradle Scripts → build.gradle, apertura el archivo y efectúa la siguiente agregación, lo cual nos permitirá mostrar imágenes directamente desde el Internet.

```
implementation 'com.squareup.picasso:picasso:2.71828'
```

2. Finalmente debes efectuar clic a **Sync Now**