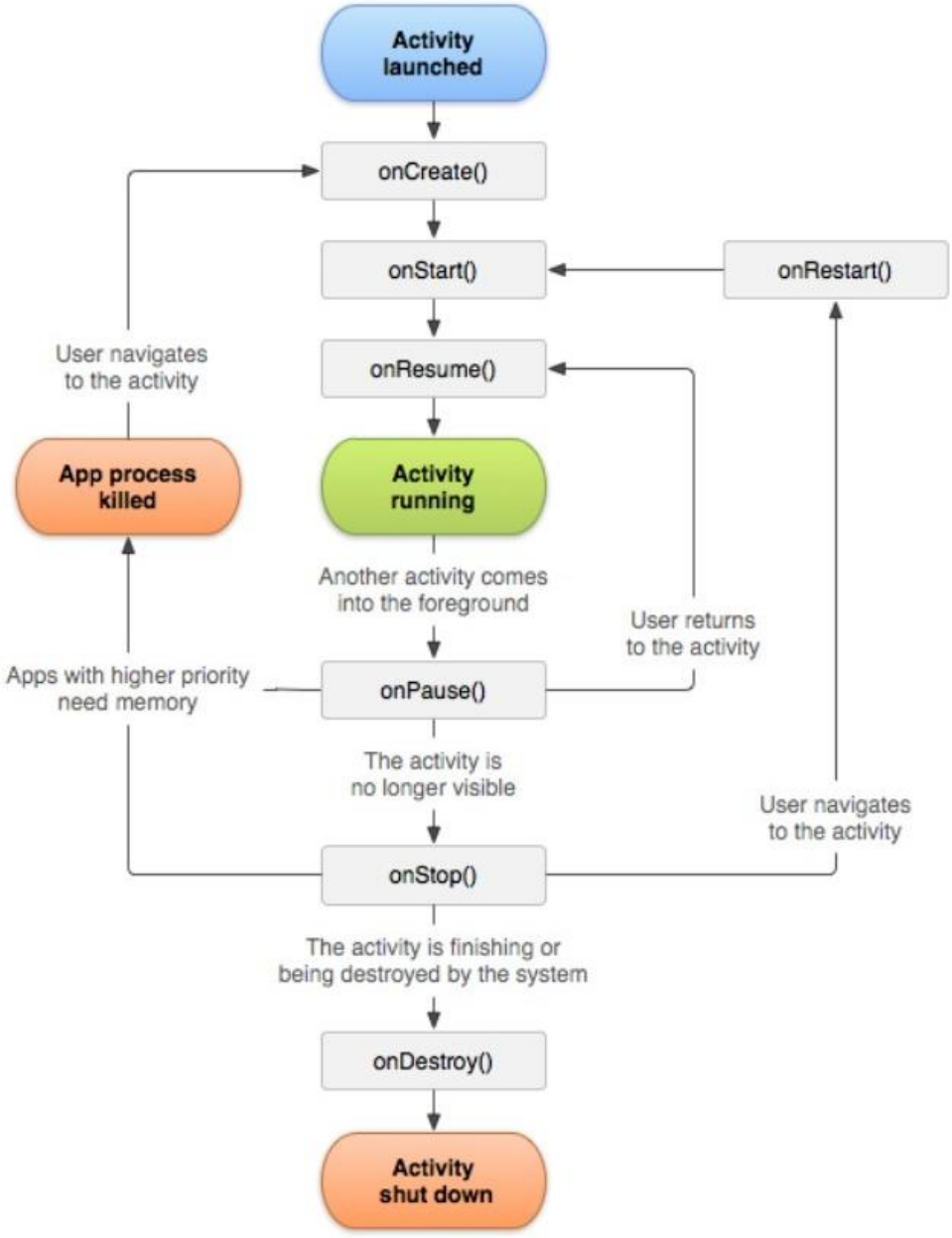
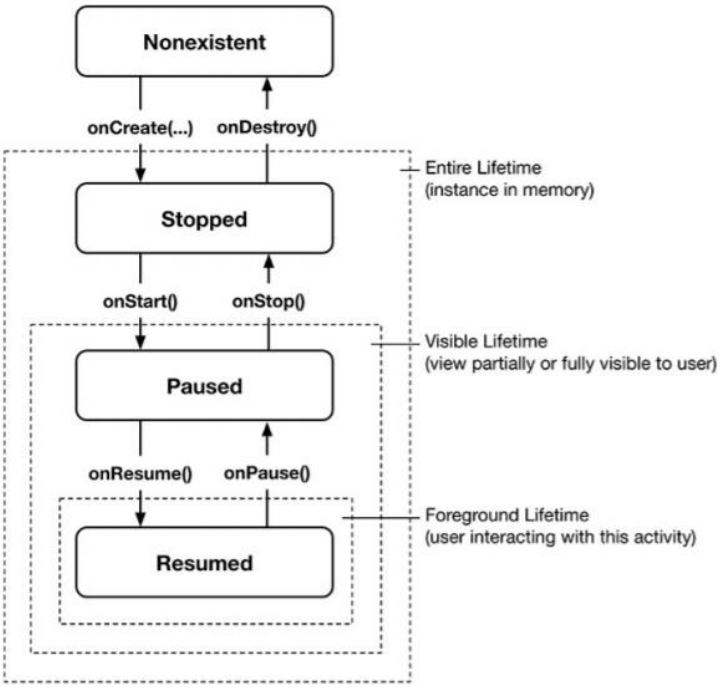
3. Ciclo de vida de una Activity

Al final del anterior capítulo, pudimos observar que, al rotar la pantalla, la aplicación nos volvía a mostrar la primera pregunta, sin importar en qué pregunta nos encontráramos. Dijimos que ese problema viene dado por el ciclo de vida de una Activity, y eso es lo que vamos a explicar en este capítulo.

Cada instancia de Activity tiene su ciclo de vida. Durante este ciclo de vida, la actividad pasa por cuatro estados: resumed, paused, stopped y nonexistent. Para cada transición entre estos estados, existe un método de la clase Activity, que notifica a la propia actividad del cambio de su estado.



Los métodos (**lifecycle callbacks**) que son llamados en la transición junto con sus estados se pueden ver detallados en el siguiente esquema:



Para saber en qué estados la aplicación se encuentra cargada en memoria, si es visible para el usuario, o si se está ejecutando en segundo plano, podemos ver la siguiente tabla.

Estado	¿En memoria?	¿Visible?	¿En 2º plano?
Nonexistent	no	no	No
Stopped	sí	no	no
Paused	sí	sí / parcialmente[1]	No
Resumed	sí	sí	sí

Para probar y ver el ciclo de vida de nuestra aplicación, podemos añadir mensajes de Log (que sólo se verán por consola) en todos los lifecycle callbacks para ver en qué momento se llaman.

Lo haremos añadiendo en cada método un mensaje, con un identificador que definiremos en la QuizActivity de la siguiente manera:

```
public class QuizActivity extends AppCompatActivity {

    private static final String TAG = "LogLifecycleCallbacks";
    private static final String KEY_INDEX = "index";
```

TAG podrá valer lo que queramos, pero se recomienda que sea algo identificativo y que no coincida con más Logs de la aplicación, como en este caso *LogLifecycleCallbacks*.

[1] Dependiendo de las circunstancias, una Activity pausada puede estar visible o parcialmente visible.

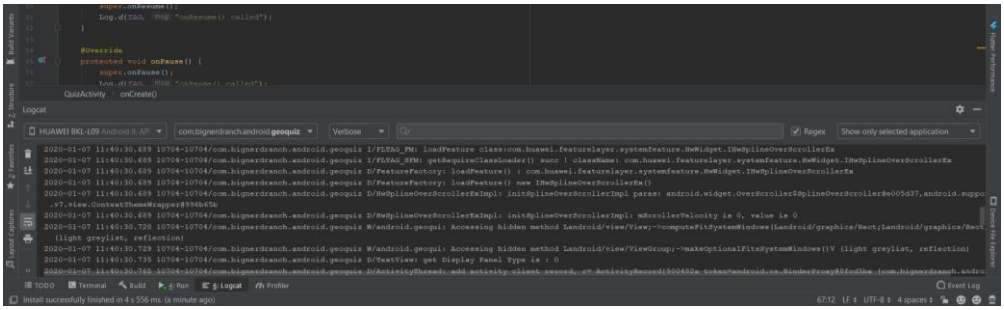
Para añadir el mensaje cada vez que se ejecute una lifecycle callback deberemos añadir esta línea en todos los métodos.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG,  msg: "onCreate(Bundle) called");
```

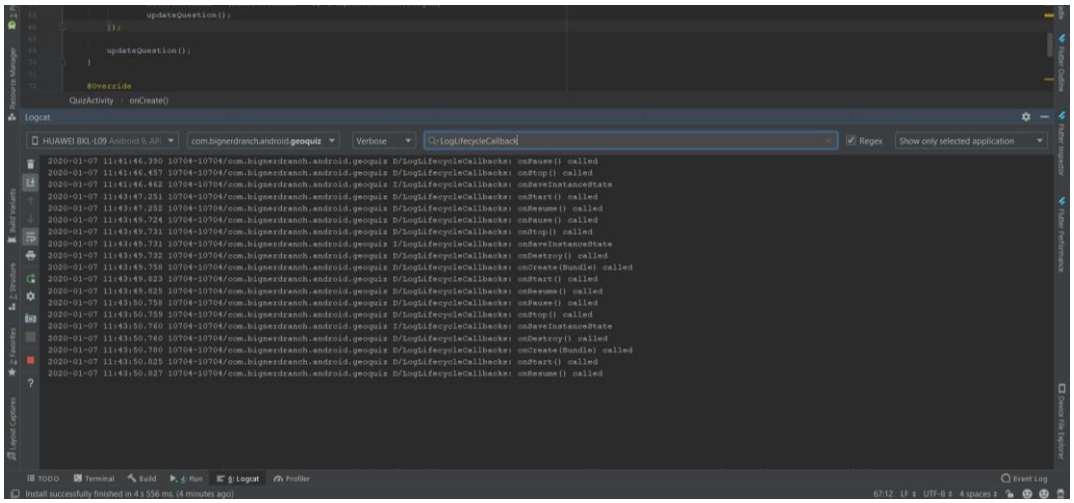
Para que aparezca en todos los métodos, indicando cuál es el que se está llamando, deberemos sobrescribir todos los lifecycle callbacks.

```
@Override
protected void onStart() {
    super.onStart();
    Log.d(TAG,  msg: "onStart() called");
}

@Override
protected void onResume() {
    super.onResume();
    Log.d(TAG,  msg: "onResume() called");
}

@Override
protected void onPause() {
    super.onPause();
    Log.d(TAG,  msg: "onPause() called");
}

@Override
protected void onSaveInstanceState(Bundle savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);
    Log.i(TAG,  msg: "onSaveInstanceState");
    savedInstanceState.putInt(KEY_INDEX, mCurrentIndex);
}

@Override
protected void onStop() {
    super.onStop();
    Log.d(TAG,  msg: "onStop() called");
}

@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d(TAG,  msg: "onDestroy() called");
}
```

Una vez hemos añadido la línea de código en la que nos mostrará por el Log de la consola el mensaje, deberemos ejecutar la aplicación y abrir el Logcat, que se encuentra en la parte inferior de la pantalla, en la pestaña adyacente a la pestaña RUN.



Una vez situados en el Logcat, tan sólo deberemos buscar el TAG definido, y sólo nos saldrán los mensajes por consola que contengan ese Log declarado.



Volviendo al bug visto en el capítulo anterior, veremos que, si rotamos la pantalla, la instancia de la QuizActivity que estábamos observando ha sido destruida y ha sido creada una nueva.

Por ahora, dejemos el bug aquí. ¡Posteriormente lo arreglaremos!

Creando un layout para la vista horizontal

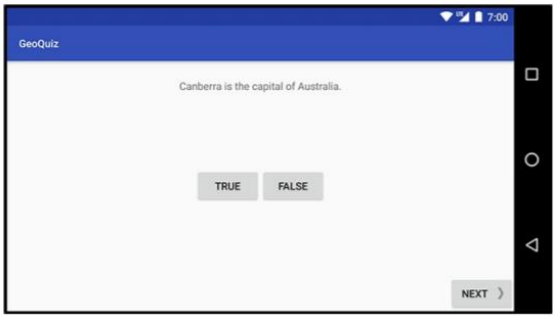
Para que la disposición de los elementos en vertical sea distinta en horizontal deberemos crear su layout. Hacemos clic derecho en el directorio *res* y seleccionamos New → Android resource directory.

Ponemos como "Directory name" *layout*, como "Resource type" ponemos *layout* y dejamos como "Source set" *main*.

Seleccionamos *Orientation* y clicamos en ">>". Finalmente nos aseguramos de que *Landscape* está seleccionado y que el nombre del directorio ahora se llama *layout-land*. Finalizamos.

El sufijo -land nos indica que esa configuración será válida cuando el dispositivo se encuentre en horizontal (landscape), y Android se encargará automáticamente de cargar ese recurso.

El nuevo layout creado (el cual podéis ver el código) para la configuración en horizontal es el siguiente:



Guardar datos antes de la rotación

Vamos a solucionar el bug de la rotación de la pantalla, el cual nos carga de nuevo la primera pregunta aunque nos encontremos en otra.

Para solucionar el bug, la QuizActivity posterior a la rotación necesitará conocer el valor del índice de la pregunta en la que nos encontrábamos antes de la rotación. La manera con la que podemos **guardar el valor del índice de la pregunta**, para después recuperarlo, es la función siguiente.

```
@Override
protected void onSaveInstanceState(Bundle savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);
    Log.i(TAG, msg: "onSaveInstanceState");
    savedInstanceState.putInt(KEY_INDEX, mCurrentIndex);
}
```

Como podemos ver, la función recibe un **Bundle, en el que deberemos guardar toda aquella información que queramos transmitir en la rotación**.

En este caso estamos guardando un dato del tipo Int: el índice. Deberemos guardarlo con un identificador (en formato String) para después poder identificarlo.

Leer datos antes de construir la Activity

Hemos visto que hemos guardado el índice de la pregunta en la que nos encontramos con el identificador KEY_INDEX, por lo que ahora deberemos recuperarlo con el mismo.

La función encargada de construir la Activity, como hemos visto hasta el momento, es la función onCreate, que también recibe un Bundle. Ese Bundle es el mismo en el que hemos guardado la información, por lo que tendremos que llamar a la siguiente instrucción para recuperar los datos.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG, msg: "onCreate(Bundle) called");
    setContentView(R.layout.activity_quiz);

    if (savedInstanceState != null) {
        mCurrentIndex = savedInstanceState.getInt(KEY_INDEX, defaultValue: 0);
    }
```

Si existe un Bundle significa que hemos guardado datos.

¡Por lo que solo tenemos que recuperarlos y listo!

Challenges

Si quieres descubrir un poquito más por tu cuenta, puedes intentar:

- Hacer que no se pueda contestar una pregunta dos veces, deshabilitando los botones de las respuestas una vez ya haya sido contestada.
- Mostrar en el Toast de respuesta correcta el porcentaje de aciertos en base al número total de preguntas existentes.