DESARROLLO DE APLICACIONES EN ANDROID

SAMSUNG

Abraham Gutiérrez Rodríguez

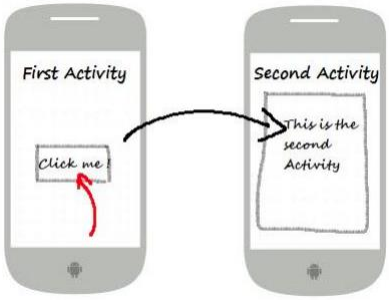
Samsung TECH INSTITUTE

Android: Conceptos Básicos
Componentes básicos

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Actividades

- La clase Activity
  - Las actividades son componentes esenciales de una aplicación, concretamente se encargan de ofrecer una pantalla con la que los usuarios pueden interactuar. Las actividades llevan asociada la interfaz de usuario.
  - De hecho, una aplicación suele estar compuesta por varias actividades que están vinculadas unas a otras de alguna forma.



Samsung TECH INSTITUTE

Android: Conceptos Básicos
Actividades

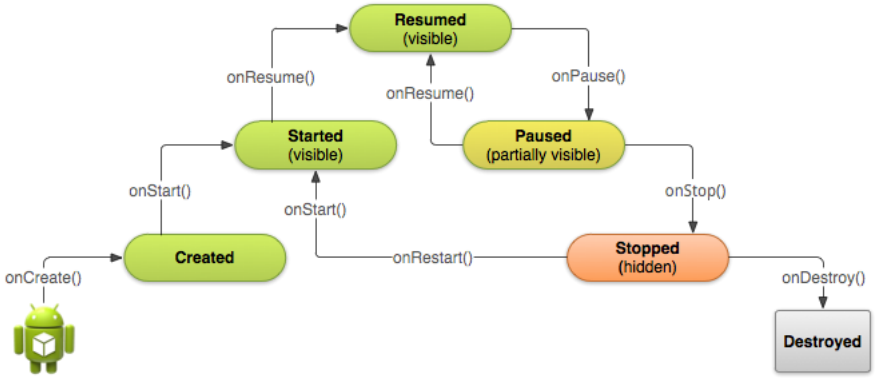
- La clase Activity
  - Generalmente, toda aplicación tiene una actividad considerada la actividad principal (*main*), la cual es la que se muestra al usuario cuando se abre la aplicación por primera vez.
  - A diferencia de otros paradigmas de programación en el que las aplicaciones se inician con un método *main()*, el sistema Android inicia el código en una instancia de una actividad mediante la invocación de métodos específicos de devolución de llamada (*callback*) que corresponden a etapas específicas de su ciclo de vida.
  - Hay una secuencia de métodos de devolución de llamada que inician una actividad y una secuencia de métodos de devolución de llamada que eliminan una actividad.

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - Durante la vida de una actividad, el sistema llama a un conjunto básico de métodos de ciclo de vida en una secuencia similar a una pirámide escalonada. Es decir, cada etapa del ciclo de vida de la actividad es una etapa separada en la pirámide.
    - A medida que el sistema crea una nueva instancia de la actividad, cada método de devolución de llamada se mueve el estado de la actividad un paso hacia la cima. La parte superior de la pirámide es el punto en el que la actividad se está ejecutando primer plano y el usuario puede interactuar con ella.
    - A medida que el usuario comienza a abandonar la actividad, el sistema llama a otros métodos que mueven el estado de actividad hacia abajo en la pirámide con el fin de desmantelar la actividad.

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad



Samsung TECH INSTITUTE

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad

```
public class Activity extends ApplicationContext {
    protected void onCreate(Bundle savedInstanceState);

    protected void onStart();

    protected void onRestart();

    protected void onResume();

    protected void onPause();

    protected void onStop();

    protected void onDestroy();
}
```

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - Dependiendo de la complejidad de la actividad, es probable que no se necesite implementar todos los métodos del ciclo de vida.
  - La implementación adecuada de los métodos del ciclo de vida de la actividad garantiza que la aplicación se comporte correctamente:
    - No se bloquea si el usuario recibe una llamada de teléfono o cambia a otra aplicación mientras se utiliza la aplicación.
    - No consume valiosos recursos del sistema cuando el usuario no esté utilizando activamente.
    - No pierde el progreso del usuario en caso de que salgan de su aplicación y vuelvan a ella en un momento posterior.
    - No se bloquee o pierda el progreso del usuario cuando la pantalla rota.

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - onCreate(): Se dispara cuando la actividad es llamada por primera vez. Aquí es donde debemos crear la inicialización normal de la aplicación, crear vistas, hacer los bind de los datos, etc. Este método da acceso al estado de la aplicación cuando se cerró. Después de esta llamada siempre se llama al onStart().
  - onRestart(): Se ejecuta cuando la actividad ha sido parada, y se quiere volver a utilizar. Después de un onStop() se ejecuta el onRestart() e inmediatamente llama a un onStart().
  - onStart(): Se ejecuta cuando la actividad se está mostrando apenas en la pantalla del dispositivo del usuario.
  - onResume(): Se ejecuta una vez que la actividad ha terminado de cargarse en el dispositivo y el usuario empieza a interactuar con la aplicación. Cuando el usuario ha terminado de utilizarla es cuando se llama al método onPause().

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - onPause(): Se ejecuta cuando el sistema arranca una nueva actividad que pasará a estar en el primer plano, pero la actividad actual sigue teniendo una parte visible. Hay que procurar que la llamada a este método sea rápida ya que hasta que no se termine su ejecución no se podrá arrancar la nueva actividad. Después de esta llamada puede venir un onResume() si la actividad que haya ejecutado el onPause() vuelve a aparecer en primer plano o un onStop() si se hace invisible para el usuario.
  - onStop(): Se ejecuta cuando la actividad ya no es visible para el usuario porque otra actividad ha pasado a primer plano. Si vemos el diagrama, después de que se ha ejecutado este método nos quedan tres opciones: ejecutar el onRestart() para que la actividad vuelva a aparecer en primer plano, que el sistema elimine este proceso porque otros procesos requieran memoria o ejecutar el onDestroy() para apagar la aplicación.
  - onDestroy(): Esta es la llamada final de la actividad, después de ésta, es totalmente destruida. Esto pasa por los requerimientos de memoria que tenga el sistema o porque de manera explícita el usuario manda a llamar este método. Si quisiéramos volver a ejecutar la actividad se arrancaría un nuevo ciclo de vida.

Android: Conceptos Básicos
Actividades

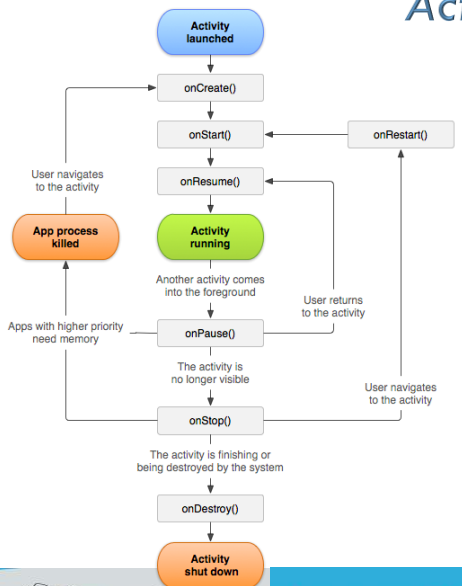
- Ciclo de vida de una actividad
  - Aunque una actividad puede pasar por diferentes estados, sólo en tres de ellos puede permanecer durante un espacio de tiempo prolongado.
    - En ejecución (resumed): En este estado, la actividad está en el primer plano y el usuario puede interactuar con ella.
    - En pausa (pauded): En este estado, la actividad está parcialmente oscurecida por otra actividad que está en primer plano es semi-transparente o no cubre toda la pantalla. La actividad en pausa no recibe la entrada del usuario y no puede ejecutar ningún código.
    - Detenido (stopped): En este estado, la actividad está completamente oculta y no visible para el usuario; se considera que está en el fondo (background). En este estado la instancia de la actividad y toda su información de estado como variables miembro se conserva, pero no puede ejecutar ningún código.

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - Los otros estados (creado e iniciado) son transitorios y el sistema se mueve rápidamente de ellos para el siguiente estado llamando al siguiente método de devolución de llamada de ciclo de vida. Es decir, después de que el sistema llama onCreate(), que llama rápidamente onStart(), que es seguido rápidamente por onResume().
  - Cuando el usuario selecciona el icono de la aplicación desde la pantalla de inicio, el sistema llama al método onCreate() de la actividad de la aplicación que se ha declarado "principal".

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad



Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - **El tiempo de vida completo** de una actividad que ocurre entre la llamada a onCreate() y la llamada a onDestroy(). Las actividades deben realizar la configuración de estado "global" (como la definición del diseño) en onCreate(), y liberar todos los recursos restantes en onDestroy().
  - Por ejemplo, si la actividad tiene un hilo de ejecución en segundo plano para descargar datos de la red, puede crear ese hilo en onCreate() y luego se detiene el hilo en onDestroy().

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - **El tiempo de vida visible** de una actividad que ocurre entre la llamada a onStart() y la llamada a onStop(). Durante este tiempo, el usuario puede ver la actividad en la pantalla e interactuar con ella.
  - Entre estos dos métodos, se deben mantener los recursos que se necesitan para mostrar la actividad al usuario.
    - Por ejemplo, se puede registrar un BroadcastReceiver en onStart() para monitorear los cambios que afectan a su interfaz de usuario, y anular el registro en onStop() cuando el usuario ya no puede ver lo que se está mostrando.
  - El sistema puede llamar onStart() y onStop() varias veces durante toda la vida útil de la actividad, siempre que la actividad alterne entre estar visible u oculta para el usuario.

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - **El tiempo de vida en primer plano** de una actividad que ocurre entre la llamada a onResume() y la llamada a onPause(). Durante este tiempo, la actividad está delante de todas las demás actividades de la pantalla y tiene el foco de entrada del usuario.
  - Una actividad puede hacer la transición del primer plano a un segundo plano con bastante frecuencia por lo que el código de estos dos métodos debe ser bastante ligero con el fin de evitar transiciones lentas que hacen que el usuario espere.
    - Por ejemplo onPause() se llama cuando el dispositivo se va a dormir o cuando aparece un cuadro de diálogo.

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - Android mantiene en memoria todos los procesos que quepan aunque éstos no se estén ejecutando.
  - Una vez que la memoria está llena y el usuario o el sistema necesite ejecutar una nueva aplicación, el sistema ha de determinar qué proceso de los que están en ejecución ha de ser eliminado.
  - Android ordena los procesos en una lista jerárquica, asignándole a cada uno una determinada "importancia". Esta lista se confecciona basándose en los componentes de la aplicación que están corriendo (actividades y servicios) y el estado de estos componentes.

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - Proceso de primer plano: (Foreground process) Hospeda una actividad en la superficie de la pantalla y con la cual el usuario está interactuando (su método onResume() ha sido llamado). Debería haber solo uno o unos pocos procesos de este tipo. Sólo serán eliminados cómo último recurso, si es que la memoria está tan baja que ni siquiera estos procesos pueden continuar corriendo.
  - Proceso visible: (Visible process) Hospeda una actividad que está visible en la pantalla, pero no en el primer plano (su método onPause() ha sido llamado). Considerado importante, no será eliminado a menos que sea necesario para mantener los procesos de primer plano.
  - Proceso de servicio: (Service process) Hospeda un servicio que ha sido inicializado con el método startService(). Aunque estos procesos no son directamente visibles al usuario, generalmente están haciendo tareas que para el usuario son importantes. El sistema siempre tratará de mantener esos procesos corriendo, a menos que los niveles de memoria comiencen a comprometer el funcionamiento de los procesos de primer plano o visibles.

Samsung TECH INSTITUTE

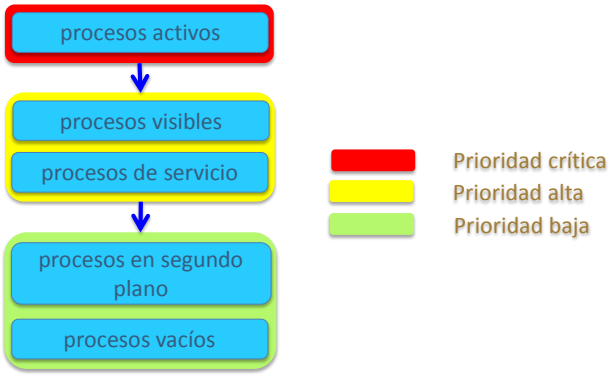
Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - Proceso de fondo o segundo plano: (Background process) Hospeda una actividad que no es actualmente visible al usuario (su método onStop() ha sido llamado). Si estos procesos son eliminados no tendrán un directo impacto en la experiencia del usuario. Generalmente, hay muchos de estos procesos.
    - Para los procesos en segundo plano, existe una lista llamada LRU (Least Recently Used). En función de esta lista se van eliminando los procesos; los primeros que se eliminan son aquellos que llevan más tiempo sin usarse. Así el sistema se asegura de mantener vivos los procesos que se han usado recientemente.
  - Proceso vacío: (Empty process) No hospeda a ningún componente de aplicación activo.
    - La única razón para mantener ese proceso es tener un "caché" que permita mejorar el tiempo de activación en la próxima vez que un componente de su aplicación sea ejecutado.

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad



Samsung TECH INSTITUTE

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - Cuando una actividad está en pausa o se detiene, se mantiene el estado de la actividad. Esto es así porque el objeto actividad aún se mantiene en la memoria y toda la información sobre sus miembros y el estado actual permanecen. Por lo tanto, todos los cambios realizados por el usuario dentro de la actividad se mantienen de modo que cuando la actividad vuelve al primer plano (cuando se "reanuda"), esos cambios están todavía allí.

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Actividades

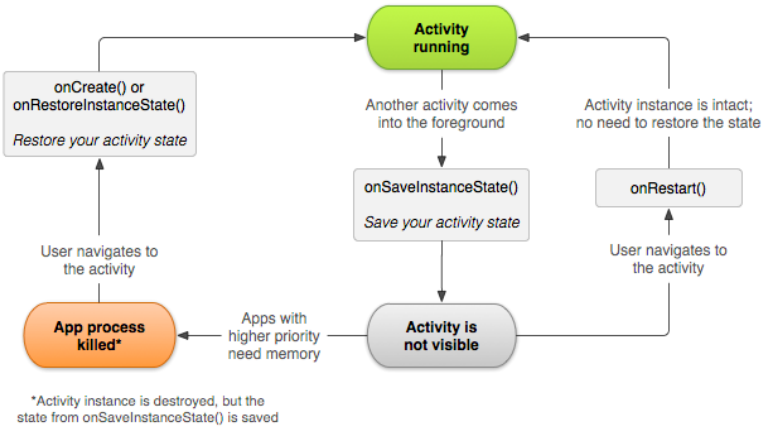
- Ciclo de vida de una actividad
  - Cuando el sistema destruye una actividad con el fin de recuperar la memoria, el objeto actividad se destruye, por lo que el sistema no puede simplemente reanudar con su estado intacto.
  - En su lugar, el sistema debe volver a crear el objeto actividad si el usuario navega de nuevo a él. Sin embargo, el usuario no es consciente de que el sistema destruye la actividad y la recreó y, por lo tanto, probablemente espera que la actividad sea exactamente como era.
  - En esta situación, se puede asegurar que la información importante sobre el estado de actividad se conserva mediante la implementación de un método de devolución de llamada adicional que le permite guardar la información sobre el estado de su actividad: onSaveInstanceState().

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad
  - El sistema llama onSaveInstanceState() antes de hacer la actividad vulnerable a la destrucción. El sistema pasa a este método un paquete (Bundle) en el que se puede guardar información de estado acerca de la actividad en forma de pares de nombre-valor, utilizando métodos como putString() y putInt().
  - Si el sistema mata a un proceso de aplicación y el usuario navega de nuevo a dicha actividad, el sistema vuelve a crear la actividad y pasa el paquete (Bundle) tanto a onCreate() como a onRestoreInstanceState().
    - El uso de cualquiera de estos métodos, puede recuperar el estado guardado en el Bundle y restaurar el estado a la actividad. Si no hay información de estado para restaurar, entonces el paquete pasado es nulo (que es el caso cuando se crea la actividad por primera vez).

Android: Conceptos Básicos
Actividades

- Ciclo de vida de una actividad



Android: Conceptos Básicos
Actividades

- Cambios de configuración
  - Si la configuración del dispositivo cambia, entonces cualquier cosa que muestra una interfaz de usuario tendrá que actualizarse para que coincida con la configuración. Debido a que la actividad es el mecanismo principal para interactuar con el usuario, incluye soporte especial para el manejo de los cambios de configuración .
    - A menos que especifique lo contrario , un cambio de configuración (por ejemplo, un cambio en la orientación de la pantalla, el idioma, los dispositivos de entrada, etc.) hará que la actividad actual sea destruida, pasando por el proceso habitual del ciclo de vida onPause(), onStop(), y onDestroy() según sea apropiado. Si la actividad estaba en el primer plano o visible para el usuario, una vez se llama a onDestroy() se creará una nueva instancia de la actividad, con la información que se haya generado a partir en la llamada a onSaveInstanceState() .
    - Esto se hace debido a que cualquier recurso de la aplicación, incluyendo archivos de diseño, puede cambiar en función de cualquier valor de configuración. Así, la única forma segura de manejar un cambio de configuración es volver a recuperar todos los recursos, incluidos los diseños, elementos gráficos y cadenas.

Android: Conceptos Básicos
Actividades

- Cambios de configuración
  - En algunos casos especiales, es posible que desee evitar el reinicio de la actividad para uno o más tipos de cambios en la configuración.
  - Esto se hace con el atributo android:configChanges del elemento <activity> en la declaración de la actividad presente en el manifiesto.
  - Para cualquier tipo de cambios en la configuración que que se registre en la declaración de la actividad se recibirá una llamada al método onConfigurationChanged() en lugar de reiniciar la actividad.
  - Si un cambio de configuración implica un elemento no registrado , la actividad será reiniciada y onConfigurationChanged() no se llamará .

Android: Conceptos Básicos
Actividades

- Coordinando actividades
  - Cuando una actividad inicia a otra, ambas experimentan transiciones del ciclo de vida. La primera actividad se pausa y se detiene (no se detendrá si tiene partes visibles), mientras que se crea la otra actividad.
  - En caso de que estas actividades compartan datos, es importante entender que la primera actividad no se detendrá completamente antes de crear la segunda. Más bien, el proceso de iniciar la segunda actividad se solapa con el proceso de detener la primera.

Android: Conceptos Básicos
Actividades

- Coordinando actividades
  - El orden de las devoluciones de llamada de ciclo de vida está bien definido, sobre todo cuando las dos actividades están en el mismo proceso y una está iniciando otra.
  - Aquí está el orden de las operaciones que se producen cuando la actividad A inicial la actividad B:
    - Se ejecuta el método onPause() de la actividad A.
    - Se ejecutan en secuencia los métodos onCreate(), onStart(), y onResume() de la Actividad de B (que recibe el foco del usuario.)
    - Entonces, si la actividad A ya no es visible en la pantalla, se ejecuta su método onStop().
  - Esta secuencia predecible de devoluciones de llamada de ciclo de vida le permite gestionar la transición de la información de una actividad a otra.
    - Por ejemplo, si tiene que escribir en una base de datos cuando la primera actividad se detiene para que la siguiente actividad pueda leerlo, entonces se debe escribir en la base de datos durante onPause() en lugar de durante onStop().

Android: Conceptos Básicos
Actividades

- Declarando una actividad
  - Todas las actividades de una aplicación deben declararse en el archivo de manifiesto con el fin de que sean accesibles para el sistema. Para declarar una nueva actividad hay que incorporar un elemento <activity> como un elemento secundario del elemento <application>.

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

Android: Conceptos Básicos
Actividades

- Declarando una actividad
  - El atributo *android:name* es el único obligatorio y especifica el nombre de la clase de la actividad.
  - Una vez publicada una aplicación, no se debe cambiar este nombre, porque de hacerse, algunas funcionalidades no estarían disponibles, tales como los accesos directos a la aplicación.
  - Hay varios otros atributos que se pueden incluir en el elemento, <activity> para definir las propiedades tales como la etiqueta de la actividad, un icono para la actividad, o un tema de estilo de interfaz de usuario de la actividad.

Android: Conceptos Básicos
Actividades

- Declarando una actividad

```
<activity android:allowTaskReparenting=["true" | "false"]
          android:alwaysRetainTaskState=["true" | "false"]
          android:clearTaskOnLaunch=["true" | "false"]
          android:configChanges=["mcc", "mnc", "locale",
                                 "touchscreen", "keyboard", "keyboardHidden",
                                 "navigation", "orientation", "screenLayout",
                                 "fontScale", "uiMode"]
          android:enabled=["true" | "false"]
          android:excludeFromRecents=["true" | "false"]
          android:exported=["true" | "false"]
          android:finishOnTaskLaunch=["true" | "false"]
          android:icon="drawable resource"
          android:label="string resource"
          android:launchMode=["multiple" | "singleTop" |
                              "singleTask" | "singleInstance"]
          android:multiprocess=["true" | "false"]
          android:name="string"
          android:noHistory=["true" | "false"]
          android:permission="string"
```

Android: Conceptos Básicos
Actividades

- Declarando una actividad
  - La actividad principal de una aplicación se define el archivo de manifiesto mediante un filtro de intención con acción MAIN y categoría LAUNCHER.

```
<activity android:name=".MainActivity" android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

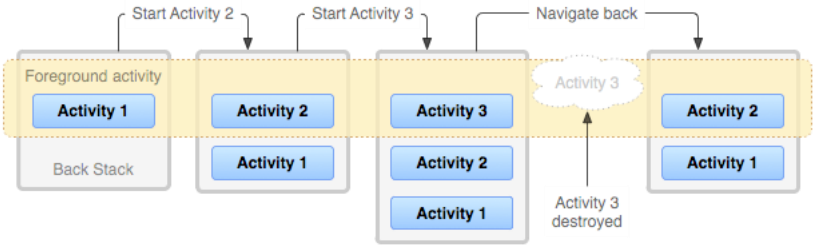
  - Si no se declaran la acción MAIN o la categoría LAUNCHER para alguna de las actividades de la aplicación, su icono no aparecerá en la pantalla de inicio de aplicaciones.

Android: Conceptos Básicos
Actividades

- Tareas
  - A pesar de que las actividades que proporcionan una funcionalidad pueden perteneces a diferentes aplicaciones, Android consigue una experiencia de usuario sin fisuras al mantener ambas actividades en la misma tarea.
  - Una tarea es un conjunto de actividades con las que los usuarios interactúan a la hora de realizar un determinado trabajo. Es una unidad cohesiva que puede pasar al “segundo plano” cuando los usuarios inician una nueva tarea o ir van la pantalla de inicio, a través del botón Inicio.
  - Las actividades están dispuestas en una pila (la "pila de vuelta atrás" o *back stack*), en el orden en que se abre cada actividad. Básicamente esta pila consiste en una pila tipo LIFO (*Last In First Out*), o lo que es lo mismo, la última actividad que fue añadida, será la primera en la cabecera de la pila.

Android: Conceptos Básicos
Actividades

- Tareas
  - Así, cuando el usuario pulse el botón atrás (Back), el sistema eliminará la actividad actual y mostrará justo la anterior en la pila.
    - Este comportamiento por defecto puede ser modificado según nuestro interés.
    - Las actividades en la pila de retroceso nunca se reorganizan.
    - Una actividad puede aparecer instanciada múltiples veces en la pila.



Android: Conceptos Básicos
Actividades

- Tareas
  - Independientemente de si una actividad se inicia en una nueva tarea o en la misma tarea que la actividad que lo empezó, en el botón Atrás siempre lleva al usuario a la actividad anterior.
  - Aunque en general no es necesario preocuparse acerca de cómo las actividades se relacionan con las tareas o cómo existen en la pila de retroceso.
  - Es posible modificar el comportamiento normal:
    - Por medio de los atributos taskAffinity, launchMode, allowTaskReparenting, clearTaskOnLaunch, alwaysRetainTaskState, y finishOnTaskLaunch del elemento <activity>
    - Por medio de banderas en las intenciones: FLAG_ACTIVITY_NEW_TASK, FLAG_ACTIVITY_CLEAR_TOP, FLAG_ACTIVITY_SINGLE_TOP…

Android: Conceptos Básicos
Actividades

- Iniciando una actividad
  - Se puede comenzar otra actividad llamando a startActivity(), y pasándole un intención que describa la actividad que desea iniciar.
    - La intención especifica o bien la actividad exacta que desea iniciar o describe el tipo de acción que desea realizar.

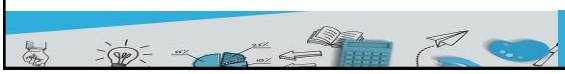
```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```

    - Una intención también puede transportar pequeñas cantidades de datos a ser usados por la actividad que se inicia.

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, recipientArray);
startActivity(intent);
```

Android: Conceptos Básicos
Actividades

- Iniciando una actividad
  - Si queremos recibir un resultado de la actividad que iniciamos llamaremos a startActivityForResult(), y pasáremos una intención que describa la actividad que se desea iniciar y un segundo parámetro entero que identifica a la llamada.
  - Cuando finaliza la actividad llamada debe ejecutar el método setResult(int) para devolver los datos al llamador. Siempre debe suministrar un código de resultado, que puede ser el resultado estándar RESULT_CANCELED, RESULT_OK, o cualquier valor personalizado a partir de RESULT_FIRST_USER. Además, opcionalmente se puede volver una intención con datos adicionales. Toda esta información aparece en el método onActivityResult() del llamador, junto con el identificador entero que se suministró originalmente en la llamada.
  - Si una actividad llamada falla por alguna razón, la actividad llamadora recibirá un resultado con el código RESULT_CANCELED.



Android: Conceptos Básicos
Actividades

- Iniciando una actividad

```
private void pickContact() {
    // Create an intent to "pick" a contact, as defined by the content provider URI
    Intent intent = new Intent(Intent.ACTION_PICK, Contacts.CONTENT_URI);
    startActivityForResult(intent, PICK_CONTACT_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // If the request went well (OK) and the request was PICK_CONTACT_REQUEST
    if (resultCode == Activity.RESULT_OK && requestCode == PICK_CONTACT_REQUEST) {
        // Perform a query to the contact's content provider for the contact's name
        Cursor cursor = getContentResolver().query(data.getData(),
        new String[] {Contacts.DISPLAY_NAME}, null, null, null);
        if (cursor.moveToFirst()) { // True if the cursor is not empty
            int columnIndex = cursor.getColumnIndex(Contacts.DISPLAY_NAME);
            String name = cursor.getString(columnIndex);
            // Do something with the selected contact's name...
        }
    }
}
```

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Actividades

- Finalizando una actividad
  - Se puede finalizar una actividad llamando a su método finish(). También puede cerrar una actividad distinta, que inició previamente, llamando a finishActivity(). En ambos casos desdpués se ejecutará el método onDestroy().
  - En la mayoría de los casos, no es necesario terminar explícitamente una actividad utilizando estos métodos. El sistema Android gestiona la vida de una actividad por lo que no es necesario terminar las propias actividades.
    - NOTA: Llamar a estos métodos podría afectar negativamente a la experiencia del usuario y sólo deben utilizarse cuando no se desea que el usuario vuelva a esta instancia de la actividad.

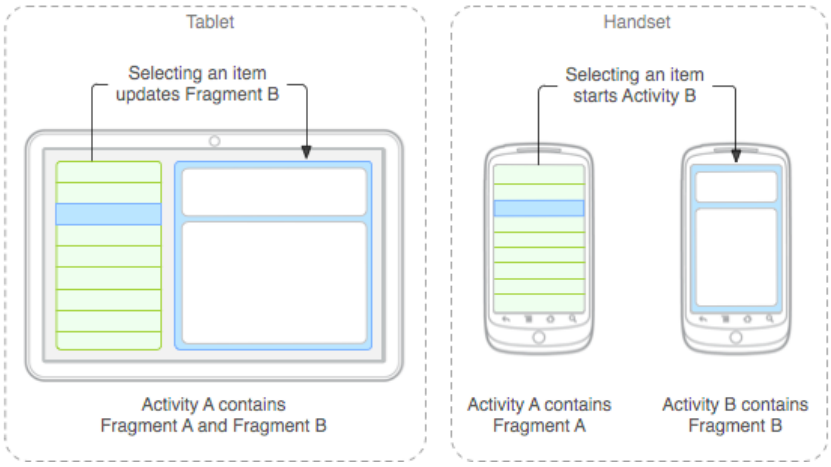
Samsung TECH INSTITUTE

Android: Conceptos Básicos
Fragmentos

- Android introdujo los fragmentos en Android 3.0 (nivel de API 11), principalmente para apoyar diseños de interfaz de usuario más dinámicas y flexibles en las pantallas grandes, como las tabletas.
  - Debido a que la pantalla de un tablet es mucho más grande que la de un teléfono, hay más espacio para combinar e intercambiar los componentes de interfaz de usuario. Los fragmentos permiten tales rediseños sin la necesidad de gestionar los cambios complejos a la jerarquía de vistas.
  - La biblioteca de soporte permite retrocompatibilidad hasta la versión Android 1.6.
- Al dividir el diseño de una actividad en fragmentos, se puede modificar el aspecto de la actividad en tiempo de ejecución y preservar esos cambios en una pila de retroceso que está administrado por la actividad.

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Fragmentos



Samsung TECH INSTITUTE

Android: Conceptos Básicos
Fragmentos

- Un Fragmento (Fragment) representa un comportamiento o una porción de interfaz de usuario en una actividad.
- Se pueden combinar múltiples fragmentos en una sola actividad para construir una interfaz de usuario de varios paneles y reutilizar un fragmento en múltiples actividades.
- Se puede pensar en un fragmento como una sección modular de una actividad, que tiene su propio ciclo de vida, recibe sus propios eventos de entrada, y que se pueden agregar o quitar mientras que la actividad está en marcha (algo así como una "subactividad" que pueda reutilizar en diferentes actividades).

Android: Conceptos Básicos
Fragmentos

- Un fragmento siempre debe estar integrado en una actividad y el ciclo de vida del fragmento se ve directamente afectado por el ciclo de vida de la actividad que lo contiene.
  - Por ejemplo, cuando se detiene la actividad, se detienen todos los fragmentos de la misma, y cuando se destruye la actividad, también se destruyen todos los fragmentos.
- Sin embargo, mientras que una actividad está en ejecución (estado resumido), se puede manipular cada fragmento de forma independiente, añadiéndolos o eliminándolos.
  - Al realizar una transacción de este tipo de fragmentos, también se puede añadir el fragmento a la pila de retroceso que está gestionado por la entrada de cada actividad pila de retroceso gestionada por la actividad. La pila de retroceso permite que el usuario invierta una transacción de fragmento (navegar hacia atrás), pulsando el botón atrás.

Android: Conceptos Básicos
Fragmentos

- Cuando se agrega un fragmento como parte del diseño de una actividad, éste vive en un ViewGroup dentro de la jerarquía de vistas de la actividad y el fragmento define su propio diseño de vista (layout).
- Se puede insertar un fragmento en el diseño de la actividad declarando el fragmento en el fichero de diseño (layout) de la actividad, mediante un elemento <fragment>, o desde el código de aplicación añadiéndolo a un ViewGroup existente.
- No es necesario que un fragmento forme parte de un diseño de una actividad; se puede usar un fragmento sin interfaz de usuario propia, como trabajador invisible para la actividad.

Android: Conceptos Básicos
Fragmentos

- Añadiendo declarativamente un fragmento:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <fragment android:name="com.example.android.fragments.HeadlinesFragment"
              android:id="@+id/headlines_fragment"
              android:layout_weight="1"
              android:layout_width="0dp"
              android:layout_height="match_parent" />

    <fragment android:name="com.example.android.fragments.ArticleFragment"
              android:id="@+id/article_fragment"
              android:layout_weight="2"
              android:layout_width="0dp"
              android:layout_height="match_parent" />

</LinearLayout>
```

Android: Conceptos Básicos
Fragmentos

- Añadiendo programáticamente un fragmento:

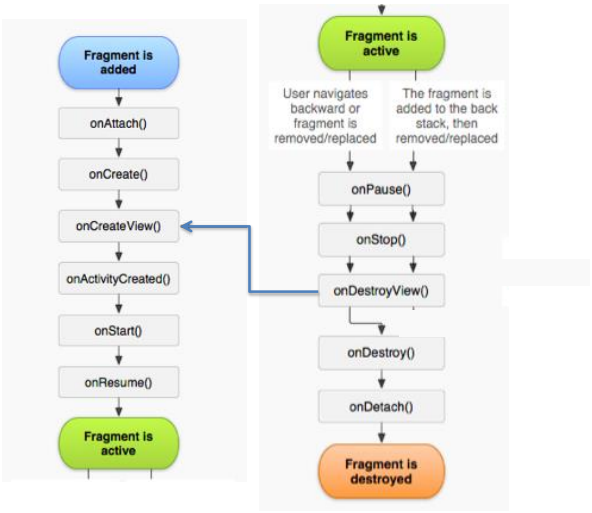
```
FragmentManager fragmentManager = getFragmentManager()
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

  - El primer argumento pasado al método add() es el ViewGroup en el que se debe colocar el fragmento, especificado por identificador del recurso, y el segundo parámetro es el fragmento que añadir.
  - Una vez que hayan realizado los cambios con FragmentTransaction, se debe llamar a commit() para que los cambios surtan efecto.

Android: Conceptos Básicos
Fragmentos

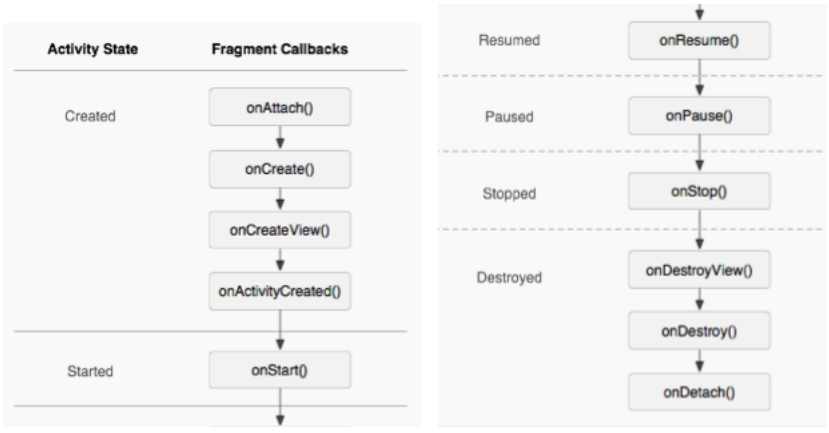
- Para crear un fragmento, se debe crear una subclase de Fragment (o una subclase existente de la misma).
- El código de la clase tiene fragmento se parece mucho al de una actividad. Contiene métodos de devolución de llamada similares a una actividad, como onCreate(), onStart(), onPause(), y onStop().
  - La conversión de una aplicación Android existente para utilizar fragmentos puede simplemente en mover el código de los métodos de devolución de llamada de la actividad a los métodos de devolución de llamada respectivos del fragmento.

Android: Conceptos Básicos
Fragmentos



Samsung TECH INSTITUTE

Android: Conceptos Básicos
Fragmentos



Samsung TECH INSTITUTE

Android: Conceptos Básicos
Fragmentos

- Por lo general, se deben implementar al menos los siguientes métodos de ciclo de vida:
  - onCreate(). El sistema ejecuta esta llamada al crear el fragmento. En ella se deberían inicializarlos componentes esenciales del fragmento que se desea conservar cuando el fragmento se reanude después de estar en pausa o parado.
  - onCreateView(). El sistema ejecuta esta llamada en el momento en que el fragmento dibuja su interfaz de usuario por primera vez. El método devuelve un objeto vista (View) que será el elemento raíz del diseño del fragmento, puede devolver *null* si el fragmento no proporciona una interfaz de usuario.
  - onPause(). El sistema ejecuta esta llamada como la primera indicación de que el usuario abandona el fragmento (aunque no siempre significa que el fragmento este siendo destruido). Es el lugar, por lo general, donde se deben realizar los cambios que tengan que persistir más allá de la sesión de usuario actual.

Android: Conceptos Básicos
Fragmentos

- Los fragmentos tienen algunas devoluciones de llamada dentro de su ciclo de vida diferentes a las actividades que se encargan de llevar a cabo acciones tales como la construcción y destrucción de la interfaz de usuario del fragmento. Estos métodos de devolución de llamada adicionales son:
  - onAttach(). Llamado cuando el fragmento se ha asociado con la actividad (se pasa la actividad como parámetro).
  - onCreateView(). Llamado para crear la jerarquía de la vista asociada con el fragmento.
  - onActivityCreated(). Informa al fragmento de que la actividad ha completado su método onCreate().
  - onDestroyView(). Se le llama cuando se está eliminando la jerarquía de la vista asociada con el fragmento.
  - onDetach(). Llamado cuando el fragmento se desvincula de la actividad.

Android: Conceptos Básicos
Fragmentos

- Hay varias subclases que es posible instanciar en lugar de la clase base Fragment:
  - DialogFragment. Muestra un cuadro de diálogo flotante. El uso de esta clase para crear un cuadro de diálogo es una buena alternativa al uso de los métodos auxiliares de diálogo en la clase de actividad, ya que se puede incorporar un diálogo fragmento en la pila de retroceso de los fragmentos gestionados por la actividad, lo que permite al usuario volver a un fragmento usado anteriormente.
  - ListFragment. Muestra una lista de elementos que son administrados por un adaptador (como un SimpleCursorAdapter), similar a ListActivity. Proporciona varios métodos para la gestión de una vista de lista, como la llamada onListItemClick() para gestionar eventos de clic.
  - PreferenceFragment. Muestra una jerarquía de objetos de preferencia en forma de lista, similar a PreferenceActivity. Esto es útil cuando se crea una actividad de "ajustes" para la aplicación.

Android: Conceptos Básicos
Fragmentos

- Añadiendo una interfaz de usuario:
  - Un fragmento se utiliza por lo general como parte de la interfaz de usuario de una actividad y contribuye con su propio diseño de la actividad.
  - Para proporcionar un diseño a un fragmento, se debe implementar el método de devolución de llamada onCreateView(), que debe devolver una vista que es la raíz de la disposición de su fragmento.
    - Nota: Si el fragmento es una subclase de ListFragment, la implementación predeterminada devuelve un ListView desde onCreateView(), por lo que no es necesario para su ejecución.
  - Para devolver un diseño (layout) desde onCreateView(), se puede inflar un recurso de diseño definido en un documento XML, haciendo uso de un objeto LayoutInflater.

Android: Conceptos Básicos

Fragmentos

- Añadiendo una interfaz de usuario:

```
public static class ExampleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.example_fragment, container, false);
    }
}
```

  - El parámetro container pasado a onCreateView() es la vista de grupo (ViewGroup) del padre en el que se inserta el fragmento de diseño. El parámetro savedInstanceState es un paquete que proporciona datos acerca de la instancia anterior del fragmento.