

Laboratorio 2



Apellidos: Moreno Vera

Nombres: Felipe Adrian

Código: 20120354I

**Asignatura: Programación en Dispositivos Móviles
(CC481)**

2016 - I

Indice

Actividad 1	(3)
Actividad 2	(6)
Actividad 3	(8)
Actividad 4	(15)
Actividad 5	(17)
Actividad 6	(18)
Actividad 7	(19)
Actividad 8	(20)
Actividad 9	(21)

Actividad 1

1. Abra el SDK Manager y describa brevemente:

1. 2 paquetes de Tools.

Android SDK Platform-tools: Son las plataformas de desarrollo a emular una versión particular de android, cuenta con ARM EABI v7^a System, Intel x86 Atom System, MIPS System y Google APIs.

Android SDK Build-tools: Es el que da y le dice a la aplicación sobre que API estamos ejecutando nuestra app, por defecto siempre esta la version de build mas reciente descargada en el SDK Manager.

2. 3 paquetes de una API

SDK platform: Es la plataforma de emulación compatible a dicha API según la arquitectura de dicha API a usar, por ejemplo, plataforma emuladora sobre Intel x86.

ARM EABI v7a System Image: Es la imagen para la arquitectura ARM en un API determinado.

Samples for SDK: Son códigos de ejemplo de aplicaciones en una determinada API.

3. 3 paquetes de Extras.















Google USB driver: Solo compatible con Windows, brinda la opción de depuración por USB cuando estamos ejecutando aplicaciones en nuestro móvil.

Android Auto Desktop Head Unit emulator: Es un emulador donde puedes ejecutar aplicaciones de Android Auto, funciona en Windows, Mac y Linux.





























Google Web driver: Es una herramienta que provee y permite la ejecución de aplicaciones web.

2. Instalar los componentes y API's según lo estudiado en teoría. No olvidar instalar en su última versión, los paquete: SDK Tools, Platforms-tools y Build-tools. Chequear siempre las actualizaciones.

Revisando Tools

)  Tools			
<input type="checkbox"/>  Android SDK Tools		24.4.1	 Installed
<input type="checkbox"/>  Android SDK Platform-tools		23.1	 Installed
<input type="checkbox"/>  Android SDK Build-tools		23.0.2	 Installed
<input type="checkbox"/>  Android SDK Build-tools		23.0.1	 Installed
<input type="checkbox"/>  Android SDK Build-tools		22.0.1	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android SDK Build-tools		21.1.2	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android SDK Build-tools		20	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android SDK Build-tools		19.1	 Installed




























Revisando el android 6.0

)  Android 6.0 (API 23)			
<input type="checkbox"/>  Documentation for Android SDK	23	1	 Installed
<input type="checkbox"/>  SDK Platform	23	2	 Installed
<input type="checkbox"/>  Samples for SDK	23	2	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android TV ARM EABI v7a System Image	23	2	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android TV Intel x86 Atom System Image	23	2	<input type="checkbox"/> Not installed
<input type="checkbox"/>  Android Wear ARM EABI v7a System Image	23	1	 Installed
<input type="checkbox"/>  Android Wear Intel x86 Atom System Image	23	1	 Installed
<input type="checkbox"/>  ARM EABI v7a System Image	23	3	 Installed
<input type="checkbox"/>  Intel x86 Atom_64 System Image	23	8	 Installed
<input type="checkbox"/>  Intel x86 Atom System Image	23	8	 Installed
<input type="checkbox"/>  Google APIs	23	1	 Installed
<input type="checkbox"/>  Google APIs ARM EABI v7a System Image	23	7	 Installed
<input type="checkbox"/>  Google APIs Intel x86 Atom_64 System Image	23	12	 Installed
<input type="checkbox"/>  Google APIs Intel x86 Atom System Image	23	12	 Installed
<input type="checkbox"/>  Sources for Android SDK	23	1	 Installed

Revisando el Android 19 y 17

<input type="checkbox"/>  Android 4.4.2 (API 19)			
<input type="checkbox"/>  SDK Platform	19	4	 Installed
<input type="checkbox"/>  <i>Samples for SDK</i>	19	6	 <i>Not installed</i>
<input type="checkbox"/>  ARM EABI v7a System Image	19	3	 Installed
<input type="checkbox"/>  Intel x86 Atom System Image	19	3	 Installed
<input type="checkbox"/>  Google APIs (x86 System Image)	19	18	 Installed
<input type="checkbox"/>  Google APIs (ARM System Image)	19	18	 Installed
<input type="checkbox"/>  <i>Glass Development Kit Preview</i>	19	11	 <i>Not installed</i>
<input type="checkbox"/>  Sources for Android SDK	19	2	 Installed
<input type="checkbox"/>  Android 4.3.1 (API 18)			
<input type="checkbox"/>  Android 4.2.2 (API 17)			
<input type="checkbox"/>  SDK Platform	17	3	 Installed
<input type="checkbox"/>  <i>Samples for SDK</i>	17	1	 <i>Not installed</i>
<input type="checkbox"/>  ARM EABI v7a System Image	17	3	 Installed
<input type="checkbox"/>  Intel x86 Atom System Image	17	2	 Installed
<input type="checkbox"/>  MIPS System Image	17	1	 Installed
<input type="checkbox"/>  Google APIs	17	4	 Installed
<input type="checkbox"/>  Sources for Android SDK	17	1	 Installed

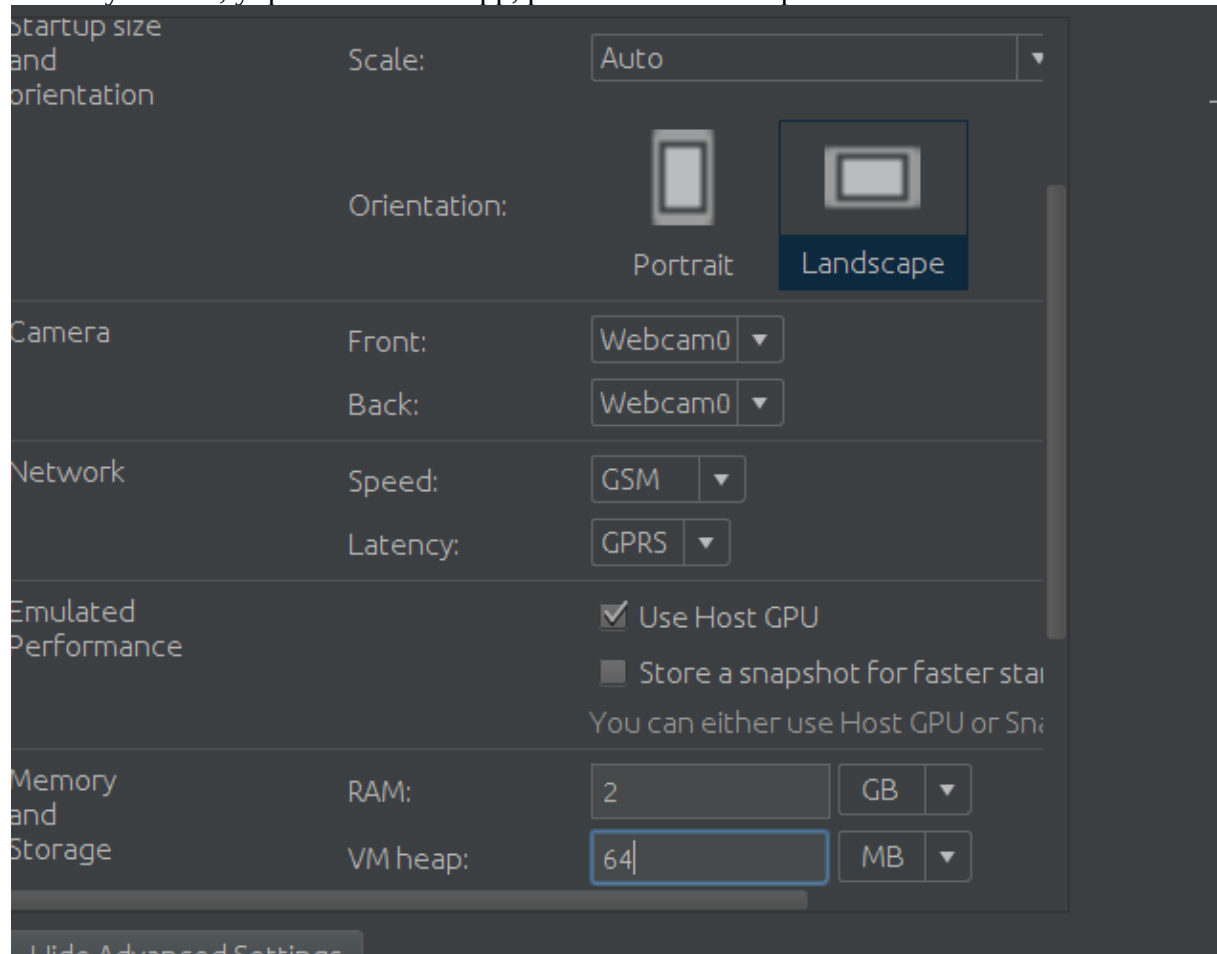
Revisando los extras.

<input type="checkbox"/>  Extras			
<input type="checkbox"/>  GPU Debugging tools		1.0.3	 Installed
<input type="checkbox"/>  Android Support Repository		25	 Installed
<input type="checkbox"/>  Android Support Library		23.1.1	 Installed
<input type="checkbox"/>  Android Auto Desktop Head Unit emulator		1.1	 Installed
<input type="checkbox"/>  Google Play services		29	 Installed
<input type="checkbox"/>  Google Repository		24	 Installed
<input type="checkbox"/>  Google Play APK Expansion Library		3	 Installed
<input type="checkbox"/>  Google Play Billing Library		5	 Installed
<input type="checkbox"/>  Google Play Licensing Library		2	 Installed
<input type="checkbox"/>  Android Auto API Simulators		1	 Installed
<input type="checkbox"/>  <i>Google USB Driver</i>		11	 <i>Not compatible wi</i>
<input type="checkbox"/>  Google Web Driver		2	 Installed
<input type="checkbox"/>  <i>Intel x86 Emulator Accelerator (HAXM inste</i>		6.0.1	 <i>Not compatible wi</i>

Actividad 2

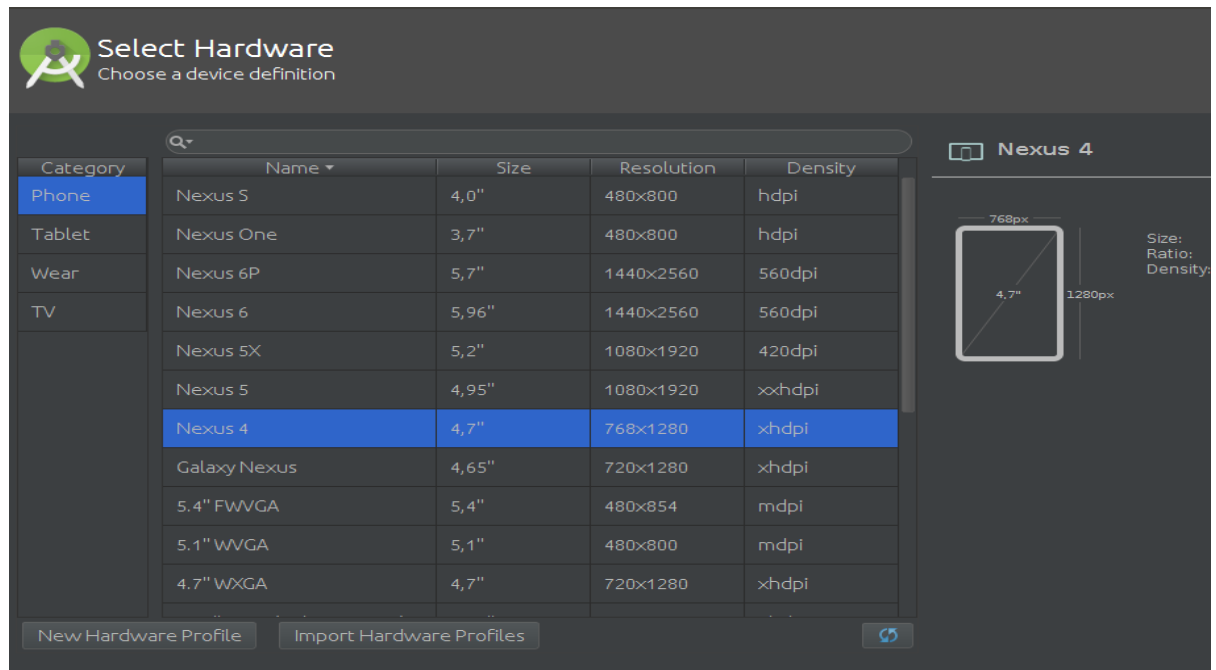
1. Edite la máquina virtual creada anteriormente. Especifique que cambios has realizado.

Se hizo cambio de las dimensiones de pantalla, la RAM, que sea capaz de emular cámara frontal y normal, y que al iniciar la app, perciba rotación de pantalla.

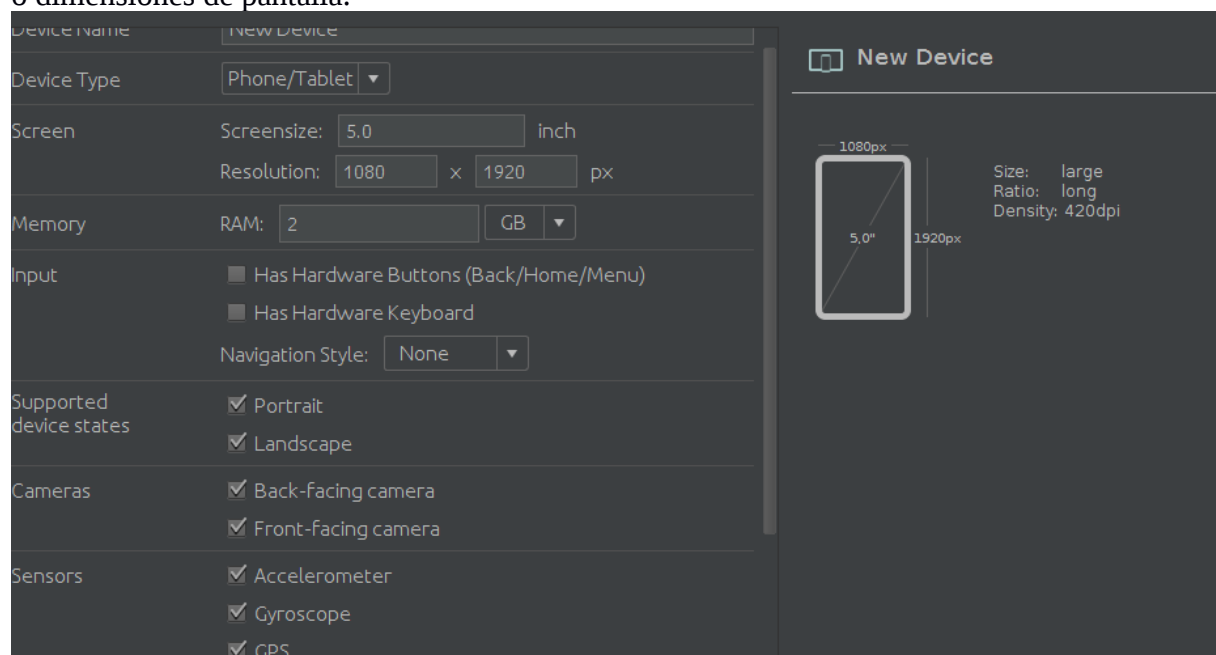


2. Realice una nueva máquina virtual y describa brevemente los parámetros que pueden configurarse de ella.

Primero nos muestra una lista de Imagenes y una lista de Hardware a escoger como ejemplo de dispositivo a emular(TV, Tablet, Wear, Phone).







También nos muestra la opción de importar un nuevo modelo de hardware (quizá en un futuro sea glass), y además crear un nuevo tipo de hardware con características como sensores o dimensiones de pantalla.



3. Es muy importante tener la máquina virtual en el nivel API de nuestro código. Cree tantas como necesites.

Como hemos instalado las APIs 17, 19 y 23, se necesitará crear una máquina virtual por cada API, pero personalmente, uso mi propio celular que está en android 4.4 y como ya he probado antes, las aplicaciones de APIs 17 y 19 se ejecutan correctamente, entonces falta crear la máquina virtual para API 23, en mi caso solo modificaré la máquina ya creada (basta cambiar el API y el tipo de hardware), he escogido Nexus 4 con Android 6.0 Marshmallow.

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Mi Nexus 4 (Edited) A...	768 x 1280: xhdpi	23	Android 6.0	x86...	716 MB	  

4. Cómo se borra la memoria RAM al iniciar la máquina virtual ?, Para qué sirve este aspecto ?.

En la imagen de arriba, donde lista nuestras máquinas, se observa una opción de run (flecha verde), un lápiz (editar) y un triángulo plomo que apunta hacia abajo.

En el triángulo plomo hacia abajo, hay una opción de “wipe data”, borra la RAM y memoria caché.

Esto sirve para que cuando inicie de nuevo el emulador de esa máquina una nueva app, la RAM solo se dedique a emular dicha app y no otras que ya había emulado (pues la máquina guarda el estado anterior, junto con todas las apps anteriores).

Actividad 3

1. Primero vemos 2 paquetes “.jar” llamados “android.jar” y “uiautomator.jar”. Busque información y explique brevemente su funcionalidad.

```
jbot@jLap: ~/Android/Sdk/platforms/android-17
jbot@jLap:~$ cd Android/Sdk/platform
bash: cd: Android/Sdk/platform: No existe el archivo o el directorio
jbot@jLap:~$ cd Android/Sdk/platform
platforms/      platform-tools/
jbot@jLap:~$ cd Android/Sdk/platforms/
jbot@jLap:~/Android/Sdk/platforms$ ls
android-17  android-19  android-23
jbot@jLap:~/Android/Sdk/platforms$ cd android-17/
jbot@jLap:~/Android/Sdk/platforms/android-17$ ls
android.jar  data          sdk.properties  source.properties  uiautomator.jar
build.prop   framework.aidl  skins           templates
```


Se ve que en esa ruta, están todas las API instaladas, y por cada API hay esos .jar android.jar : Cuando instalamos los SDK platforms por cada API, en android.jar es el empaquetado que justamente es el SDK Platform, contiene librerías de cada tipo de imagen para dicha API, entonces el android.jar contiene los java byte code, apara cada imagen a ejecutar en la máquina virtual.

uiautomator.jar: Es una herramienta que testea automáticamente la UI de Android(en este caso, la respectiva API), cuando iniciamos el emulador se genera todo el sistema android.

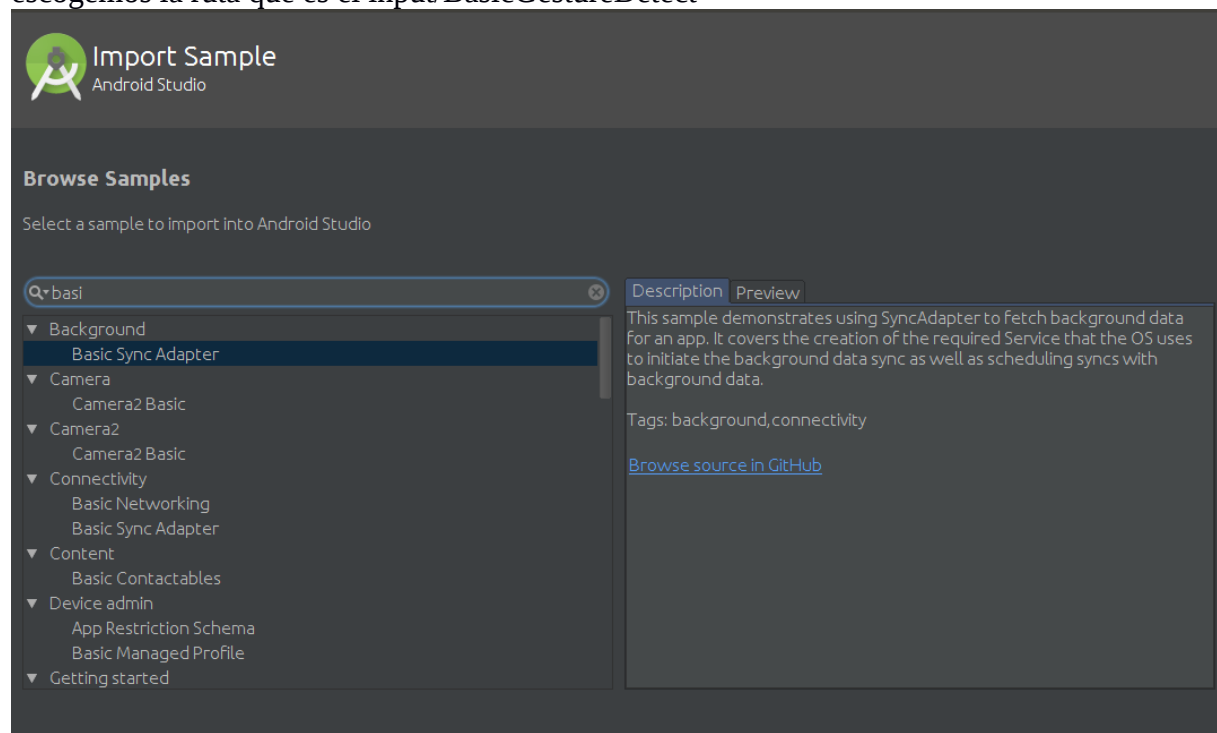
2. Otro directorio de interés es el de “Examples” que como hemos visto contiene una serie de ejemplos que podemos ejecutar en Android.

Ejecute un ejemplo de estos y describa brevemente los pasos para ello.

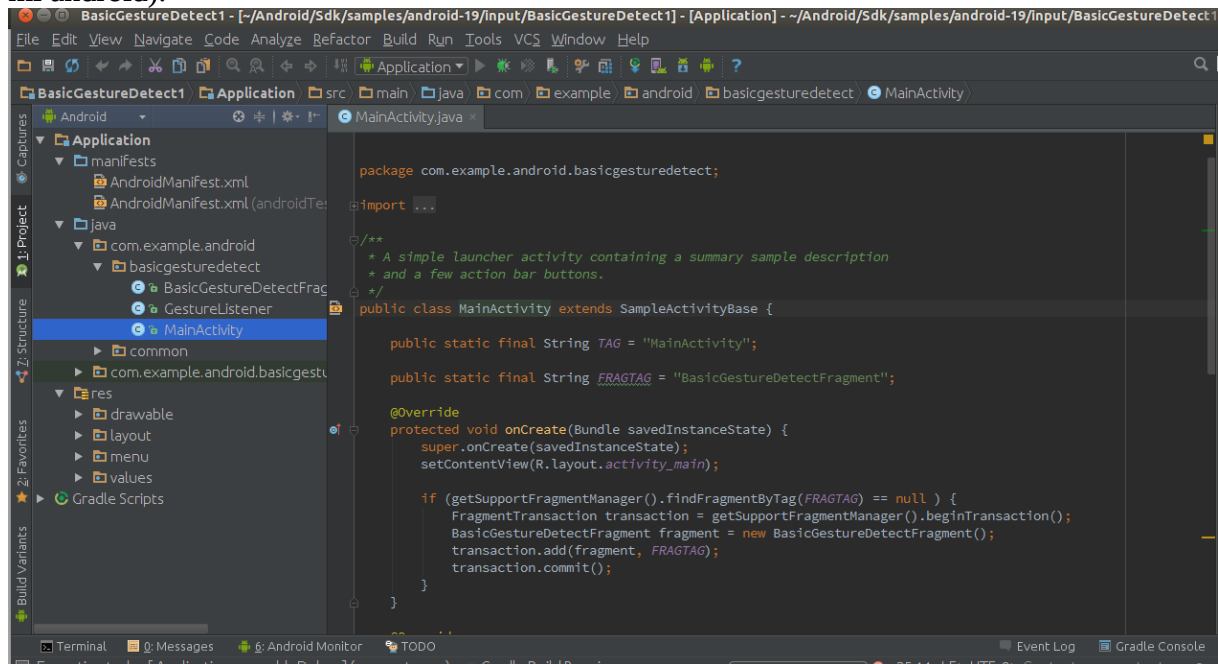
Primero vamos a instalar algún example del SDK Manager, escogeré la API 19.

Android 4.4.2 (API 19)			
<input type="checkbox"/> SDK Platform	19	4	Installed
<input checked="" type="checkbox"/> Samples for SDK	19	6	<input type="checkbox"/> Not installed

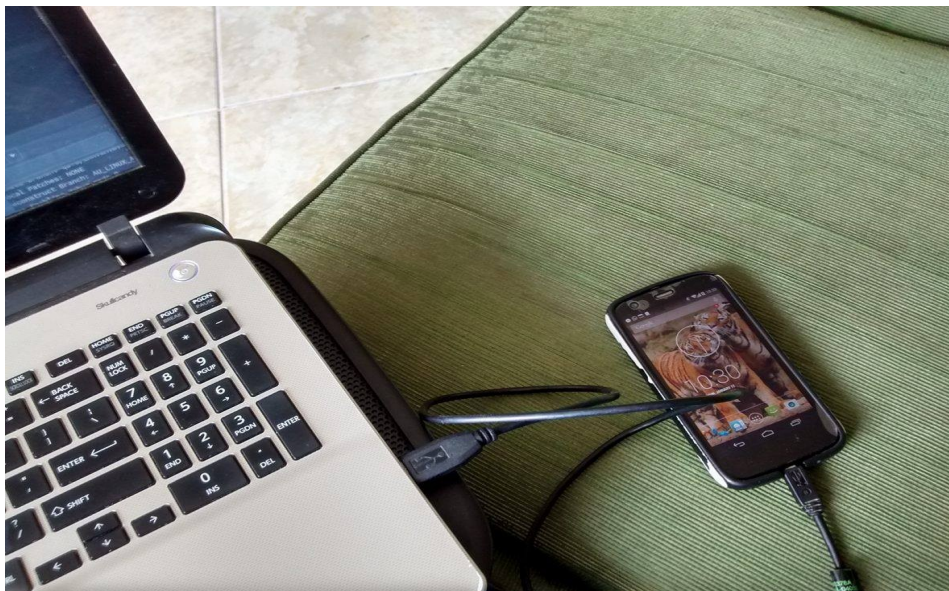
Abrimos el proyecto de la API 19 instalado, vamos a File ->New -> Import sample, escogemos la ruta que es el input/BasicGestureDetect



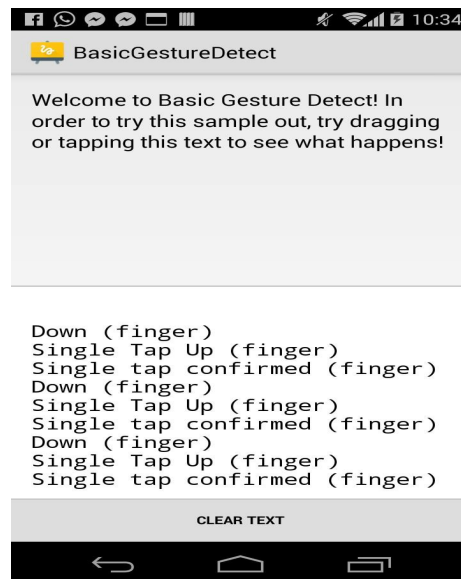
Listo lo tenemos importado ahora lo ejecutamos en nuestra máquina(en mi caso lo ejecuto en mi android).



Ejecutando en mi móvil.



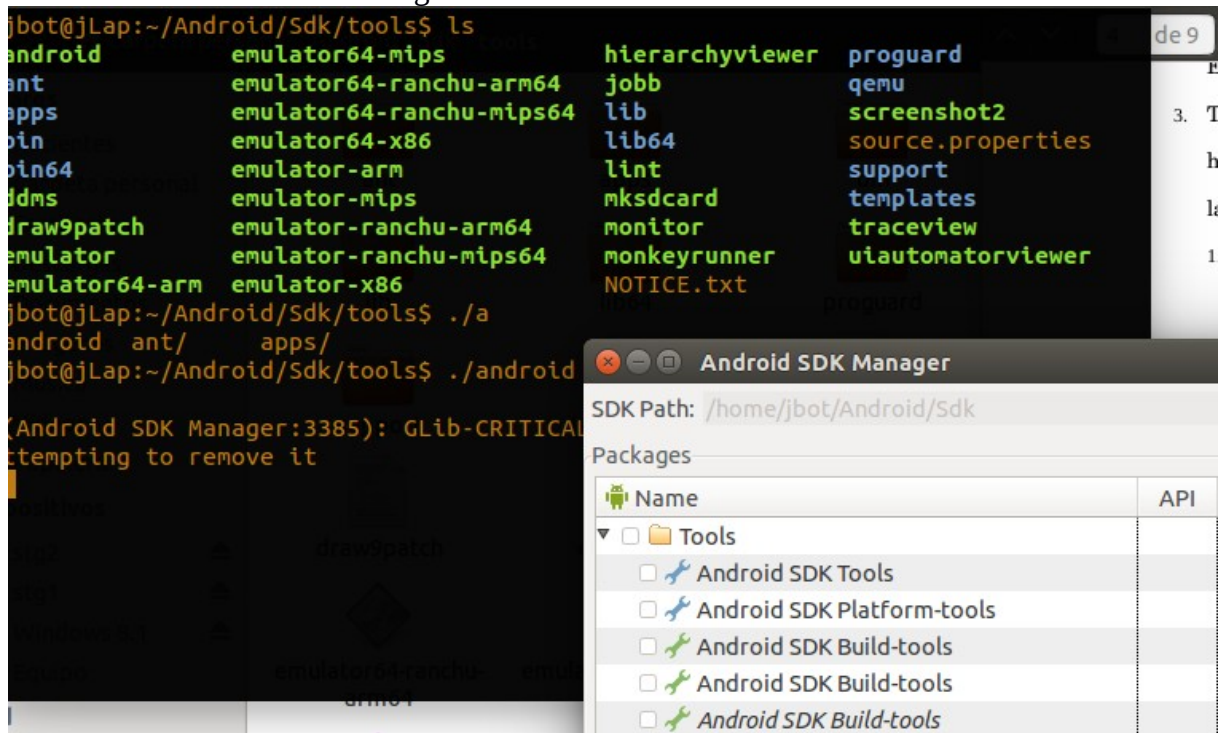
La app lanzada en mi móvil.



3. También observamos el paquete Tools que contiene una serie de herramientas clasificadas en 2 grupos como hemos visto. Veamos las Herramientas SDK:

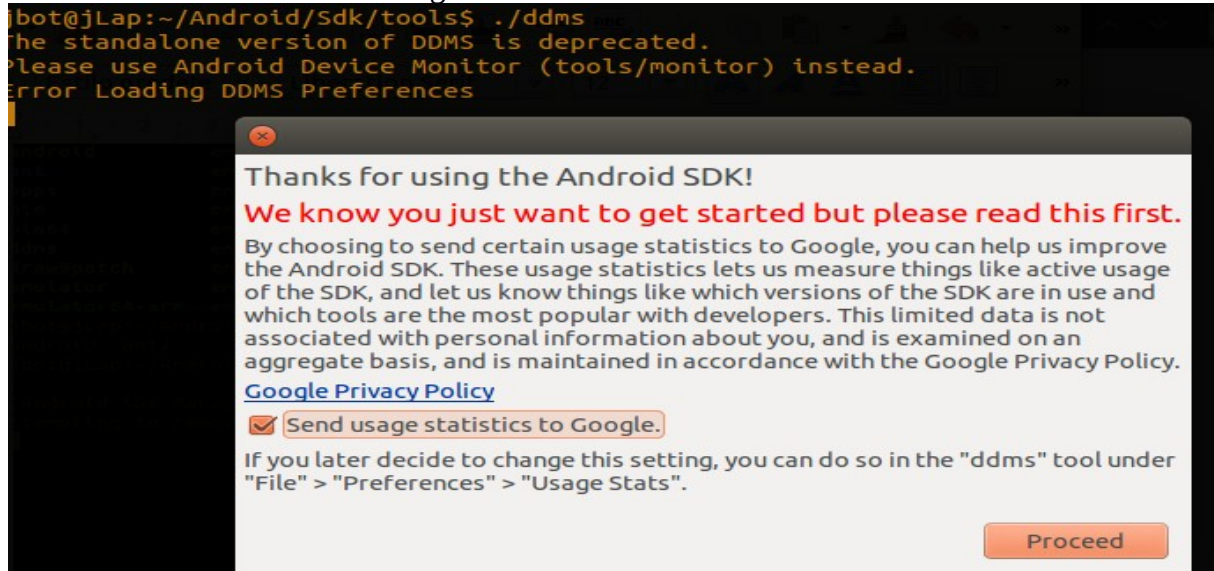
1. android: que permite gestionar máquinas virtuales, proyectos y componentes a nuestra máquina virtual. Ejecútelo haciendo doble click. Qué nos aparece?

Nos lanza el SDK Manager.

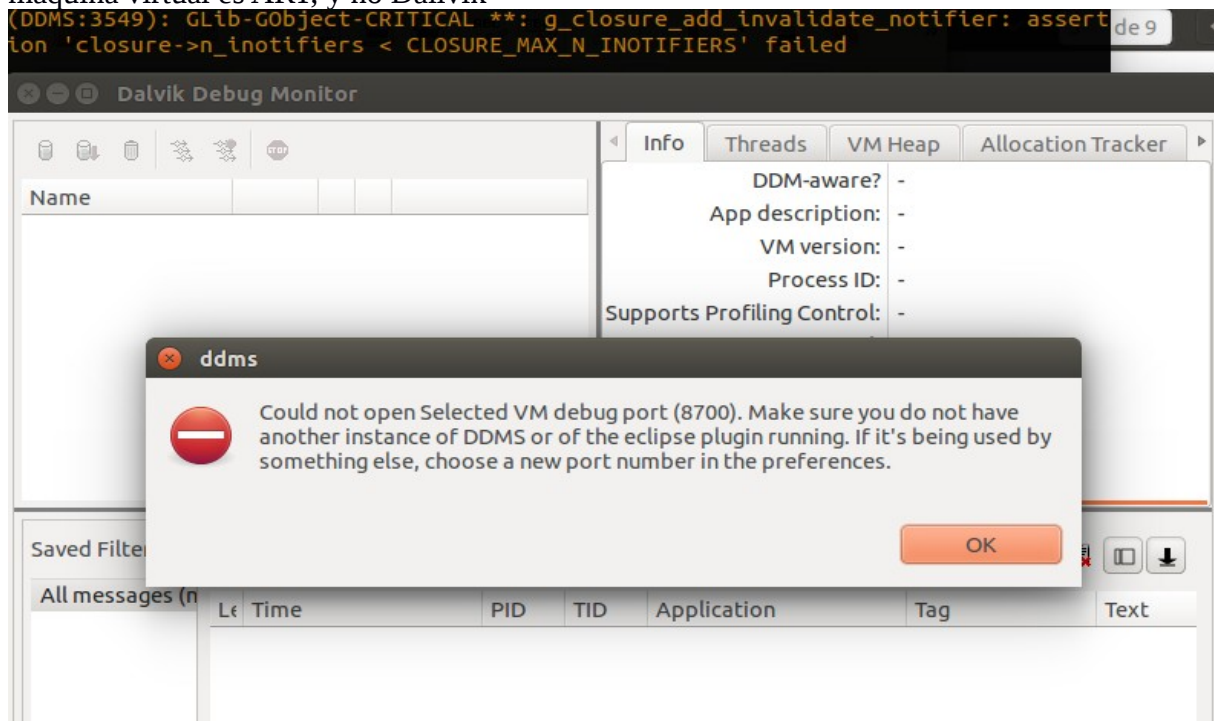


2. ddms: permite depurar aplicaciones Android. Haga también doble click para ejecutarlo. Explique la salida dada.

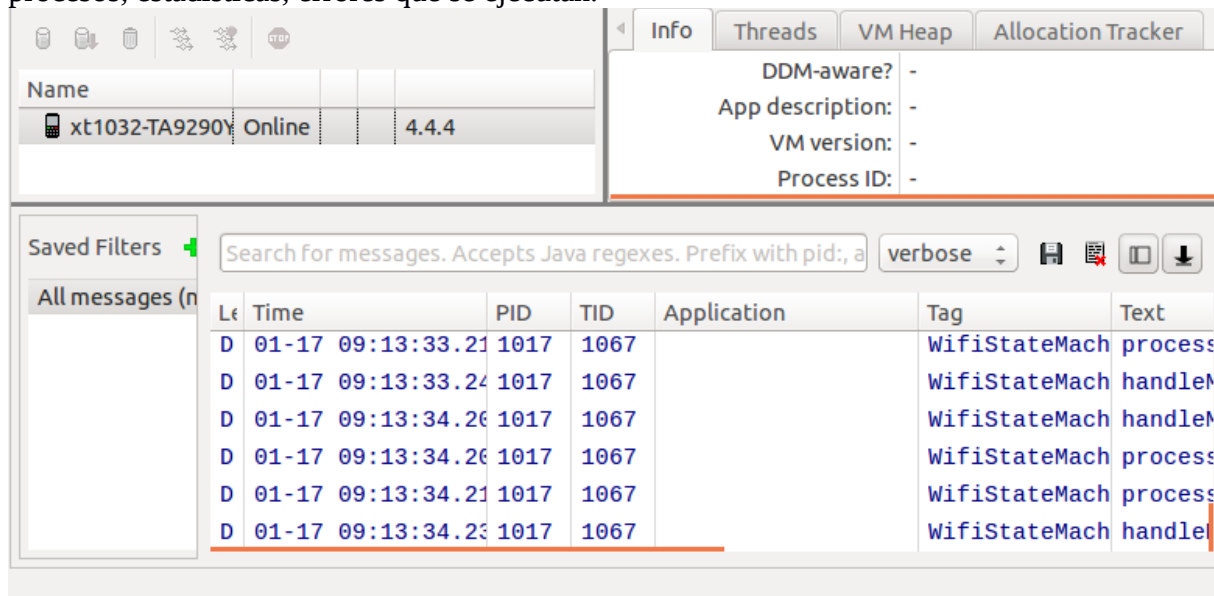
Nos abre el Dailvik Debug Monitor



Al momento de iniciar el ddms, nos bota error de que no se reconoció la máquina o que hay un proceso en Eclipse(pero no es mi caso, pues no tengo eclipse ni instalado) y además mi máquina virtual es ART, y no Dailvik



Ahora conectando mi dispositivo motorola con máquina Dailvik. Aparece y muestra los procesos, estadísticas, errores que se ejecutan.

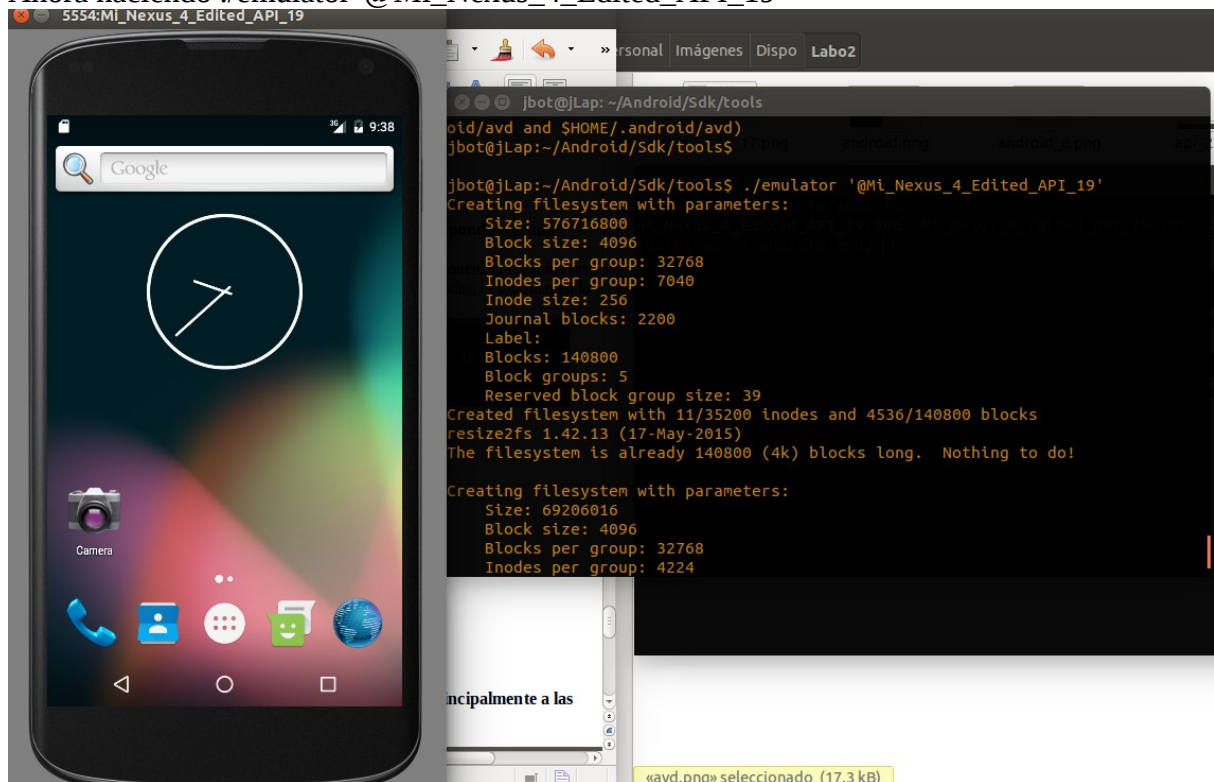


3. Emulador: permite ejecutar emulados Android que corresponde con una arquitectura.

Para ejecutar el emulador nos piden un nombre de máquina, entonces la máquina que cree será la que use, aparte, TODAS las máquinas virtuales creadas, se guardan en la carpeta `.android/avd`, haciendo `ls ./.android/avd` obtenemos:

```
jbot@jLap:~/.android/avd$ ls
Mi_Nexus_4_Edited_API_19.avd  Mi_Nexus_4_Edited_API_19.ini
jbot@jLap:~/.android/avd$
```

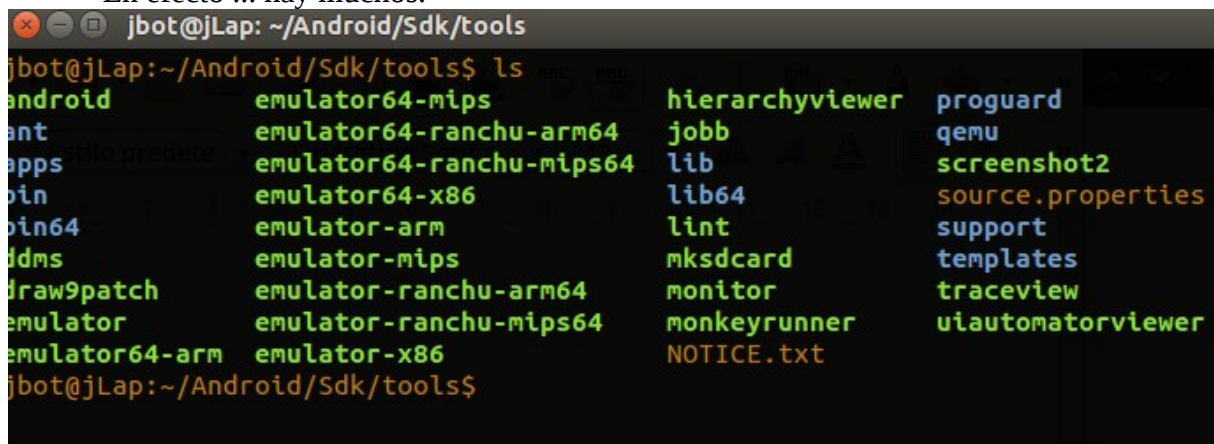
Ahora haciendo `./emulator '@Mi_Nexus_4_Edited_API_19'`



Vemos que lanzamos nuestra máquina virtual :D

4. Como vemos hay muchos paquetes que corresponden principalmente a las herramientas instaladas en nuestro SDK.

En efecto ... hay muchos.



4. Herramientas de plataforma: La más importante es adb. Explique brevemente para qué sirve esta herramienta.

adb: Android Debug Bridge, se encuentra en SDK/platforms-tools. es una herramienta de consola que le permite comunicarse con una instancia de emulador o dispositivo con Android conectado. Es un programa cliente-servidor que incluye tres componentes:

Un cliente, que se ejecuta en el equipo de desarrollo. Puede invocar un cliente desde un shell mediante la emisión de un comando adb. Otras herramientas de Android como DDMS también crean clientes adb.

Un servidor, que se ejecuta como un proceso en segundo plano en el equipo de desarrollo. El servidor gestiona la comunicación entre el cliente y el demonio adb se ejecuta en un emulador o dispositivo.

Un demonio, que se ejecuta como un proceso en segundo plano en cada emulador o dispositivo instancia.

Actividad 4

1. Creamos un nuevo proyecto en blanco.

Creamos New-> Project... Empty Blank, damos nombre de activity y Layout, finish.

2. Declaramos un String de este tipo en el código Java de nuestra actividad con el valor "Programación". No le establecemos ningún modificador.

```
package com.example.jbot.lab2_activ4;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    String cadena = "Programación";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

3. Que significado tiene static y final? En un dispositivo móvil es muy importante porque esto reduce el procesamiento, en las computadoras no tiene gran relevancia.

Static: En ese caso la variable es única para todas las instancias (objetos) de la clase (ocupa un único lugar en memoria). A veces a las variables de clase se les llama variables estáticas. Si no se usa static, el sistema crea un lugar nuevo para esa variable con cada instancia (la variable es diferente para cada objeto). En el caso de una constante no tiene sentido crear un nuevo lugar de memoria por cada objeto de una clase que se cree. Por ello es adecuado el uso de la palabra clave static.

Final: En este contexto indica que una variable es de tipo constante: no admitirá cambios después de su declaración y asignación de valor. final determina que un atributo no puede ser sobrescrito o redefinido, es decir, es como una constante. Toda constante declarada con final ha de ser inicializada en el mismo momento de declararla. final también se usa como palabra clave en otro contexto: una clase final (final) es aquella que no puede tener clases que la hereden.

Nota extra: Cuando usamos “static final” se dice que creamos una constante de clase, un atributo común a todos los objetos de esa clase.

4. Qué son los modificadores de acceso ? Que valor le darías ?

Un modificador de acceso es una palabra reservada (public, private, protected) que restringe el acceso a una variable o clase(a ser instanciada) por contextos de otras clases.

Además, cuando no le damos modificador, la variable puede ser vista o accedida por todas las clases del mismo package.

A la variable String cadena = “Programación” no le daría valor de modificador, pues puede que sea usada sólo en la clase del Activity main(por ahora).

MODIFICADOR	CLASE	PACKAGE	SUBCLASE	TODOS
public	Sí	Sí	Sí	Sí
protected	Sí	Sí	Sí	No
No especificado	Sí	Sí	No	No
private	Sí	No	No	No

Actividad 5

1. En el mismo proyecto creamos una clase en nuestro paquete Java.

Creamos la clase “claseEjemplo”.

2. Creamos un String que se llame Dato.

3. Creamos el constructor de nuestra clase. La dejamos de momento vacía.

4. Creamos un método que se llame ejecutar. La dejamos de momento vacía.

5. Creamos los métodos setData() y getData(). Recuerda la funcionalidad.

setData() servirá para modificar el valor de la variable Dato, y getData() será para que nos retorne dicho valor, entonces, con estas funciones, la variable String Dato debe tener el modificador “private”.

```
/**
 * Created by jbot on 17/01/16.
 */
public class claseEjemplo {
    String Dato; // variable tipo String
    public claseEjemplo(){} // constructor
    void ejecutar(){} // metodo ejecutar
    void setData(){} // metodo setData
    String getData(){
        return "a";
    } // metodo getData
}
```

6. IMPORTANTE: En android no es buena práctica los get y set. Por qué ? Porque en Android cada vez que se lance get y set hace ejecutar más cosas lo que consume más procesamiento. Por lo tanto, intentamos omitir dichos métodos.
Escribe una alternativa.

Una manera de hacer esto, es usar el fichero String, que contiene las variables creadas, con contenido, de manera que mediante las funciones getString(R.string.#) donde # es la etiqueta de la variable registrada en String.xml, obtenemos el valor, y para hacer el set, mediante un TextView, y su método setText(#), donde # es el nombre de la variable tipo String, le damos el valor a mostrar en la app.

Actividad 6

1. Qué significa que le pongamos modificador **public** ?
2. Qué significa que le pongamos modificador **protected**?
3. Qué significa que le pongamos modificador **private**?
4. Qué nivel es mas seguro ?

MODIFICADOR	CLASE	PACKAGE	SUBCLASE	TODOS
public	Sí	Sí	Sí	Sí
protected	Sí	Sí	Sí	No
No especificado	Sí	Sí	No	No
private	Sí	No	No	No

Según visto en la actividad 4 ...

1. Si le ponemos modificador **public** a una variable, ésta será visible o accesible por cualquier clase y subclase(heredada) y todas las demás clases dentro del package o proyecto.
2. Si le ponemos modificador **protected**, la variable será visible o accesible por el package (en este caso, el package es la carpeta en donde se encuentra nuestra clase que contiene dicha variable, puede ser que en nuestro proyecto creamos una carpeta Meowpack, entonces una variable que este en una clase dentro de esa carpeta con modificador **protected**, solo será visible por las clases dentro de esa carpeta Meowpack).
3. Si le ponemos modificador **private**, esa variable solo será visible o accesible por la clase contenedora de dicha variable, mas ninguna otra podrá acceder o modificar dicha variable.
4. A nivel de seguridad de variable, si hablamos de modificar, leer o escribir, el nivel de modificador más seguro es **private**, pues solo la clase contenedora tiene acceso a dicha variable.

Actividad 7

1. Creamos una nueva clase de nuestro paquete, por ejemplo, “Modelo”.
2. Creamos una cadena con modificadores final static y un valor que nosotros queramos, por ejemplo, “db_usuarios_acceso”.
3. Creamos otro modificador igual con valor “col_nombre_usuario”.
4. Que funcionalidad tiene final y static? Por que son importantes en android?

Como la NOTA que puse en la actividad 4, Cuando usamos “static final” se dice que creamos una constante de clase, un atributo común a todos los objetos de esa clase. Es importante para que en todo package sea una constante.

```
* Created by jbot on 17/01/16.
*/
public class Modelo {
    final static String cadenaMod1 = "db_usuarios_acceso";
    final static String cadenaMod2 = "col_nombre_usuario";
}
```

5. Volvemos a “ClaseEjemplo” y creamos un String que se vaya a almacenar dándole el valor “Modelo.nombreString1”. Que hemos realizado con esto? Tenga en cuenta que crear objetos es una carga pesada y para los dispositivos móviles más.

```
public class ClaseEjemplo {
    String dato; // variable tipo String
    String alman = "Modelo.nombreString1";
    public ClaseEjemplo() {} // constructor
    void ejecutar() {} // metodo ejecutar
    void setData() {} // metodo setData
    String getData(){
        return "a";
    } // metodo getData
}
```

Estamos creando la variable alman para que tenga como valor una referencia de la clase Model y que apunte a una variable nombreString1, que de momento no se sabe cual es.

Actividad 8

1. Creamos una clase privada dentro de ClaseEjemplo.
2. Creamos un método por ejemplo acción que haga referencia al método ejecutar() que tiene solamente la sentencia: `String var = dato;`

```
* Created by jbot on 17/01/16.
*/
public class ClaseEjemplo {
    String dato; // variable tipo String
    String alman = "Modelo.nombreString1";
    public ClaseEjemplo(){} // constructor
    private class Ejemplito {
        Ejemplito(){}
        void accion() {
            ejecutar();
        }
    }
    void ejecutar(){
        String var = dato;
    } // metodo ejecutar
    void setData(){} // metodo setData
    String getData(){
        return dato;
    } // metodo getData
}
```

3. Lo puede hacer y funciona pero también consume muchos recursos por lo que nunca puede tener en una clase privada este planteamiento. Exponga una solución.

Para el llamado del método de Ejemplito sería, Ejemplito A.accion(), el cual llama al método del padre ejecutar, y si, consume muchos recursos porque usa clases internas.

Una solución sería que creamos el método ejecutar dentro de la clase privada, ya que solo en este caso esta igualando `String var = dato`, pero aún así consume al instanciar la clase privada para lanzar ejecutar().

4. Por tanto nunca cree clases privada que llamen a métodos de clases públicas.

Ok, nunca más crearé clases que llamen métodos, por que consumen recursos. :D

Actividad 9

1. Borramos la clase privada creada anteriormente.
2. Hacemos un for normal de 1 a 10.
3. Estos for deberán ser sustituidos siempre que se pueda por `for(String s:string)` ya que el rendimiento se mejora al conocer el tamaño de antemano, por lo que minimiza el procesamiento.
4. También es importante la utilización de `if` antes de los `for` para no realizar recorridos innecesarios. Como veis se incide bastante en minimizar el procesamiento en los móviles.
5. un `if` tentativo podría ser: `if(string.size()>0)`

Se creó una pequeña función para que no fastidie mostrando que es ilegal definir un `for`.

Y además para se usó la colección de `Integers` en vez de `Strings` como se muestra.

Se usa `size` cuando se usa colecciones, y `length` para `String`.

`alman.length()` y `Numeros.size()`

```
public class claseEjemplo {
    String Dato; // variable tipo String
    String alman = "Modelo.nombreString1";
    ArrayList<Integer> Numeros;
    public claseEjemplo(){} // constructor
    void ejecutar(){
        String var = Dato;
    } // metodo ejecutar
    void setData(){} // metodo setData
    String getData(){
        return Dato;
    } // metodo getData

    void menuFor() {
        if (Numeros.size() > 0 && alman.length()>0) {
            for (Integer a : Numeros) {
                i
            }
            for (int i = 0; i < 10; i++) {
                i
            }
        }
    }
}
```

Referencias

https://en.wikipedia.org/wiki/Application_programming_interface
http://www.htcmania.com/mediawiki/index.php/Programaci%C3%B3n_de_aplicaciones_para_m%C3%B3viles_Android_-_Unidad_1#Las_versiones_de_Android_y_niveles_de_API
https://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android
<https://joefebrian.wordpress.com/2012/10/23/how-to-find-jdk-version-on-ubuntu-12-04/>
<http://developer.android.com/intl/es/sdk/installing/index.html?pkg=studio>
<http://developer.android.com/intl/es/tools/support-library/index.html>
<http://definicion.de/dispositivo/>
<https://infinum.co/the-capsized-eight/articles/art-vs-dalvik-introducing-the-new-android-runtime-in-kit-kat>
<http://developer.android.com/intl/es/guide/topics/providers/content-provider-creating.html#ContentProvider>
<http://developer.android.com/intl/es/tools/help/adb.html>
http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=665:public-private-y-protected-javatipos-de-modificadores-de-acceso-visibility-en-clases-subclases-cu00693b&catid=68:curso-aprender-programacion-java-desde-cero&Itemid=188
<http://stackoverflow.com/questions/10101174/how-to-get-and-set-string-resource-values-in-android> – set y get Android.