Lecture 22: Android Development

Instructor: Mitch Neilsen

Office: N219D

Quote of the Day

Outline

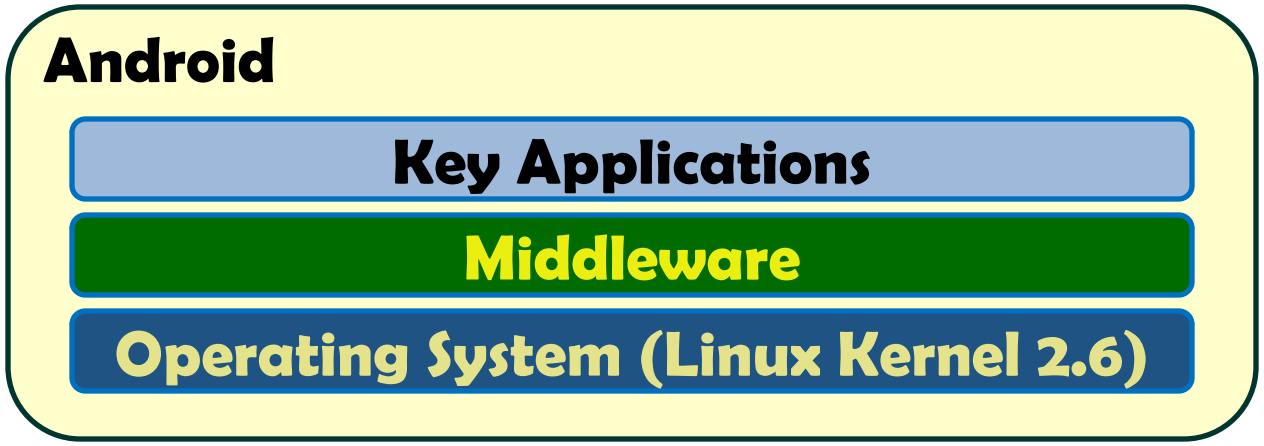
- Android background
- How to install/use tools
- Create
    - Applications

      - See: http://developer.android.com
    - Use Activities and Services in an application
    - Content providers
    - Broadcast receivers

Android

A software stack for mobile devices developed and managed by the Open Handset Alliance (OHA).

Open source software under an Apache License.



Open Handset Alliance (OHA)

What is Android?

Created in 2003 by Andy Rubin

- Google purchased in 2005

Linux-based kernel for the ARM architecture

- Dalvik Java Virtual Machine

  - register based (less memory)

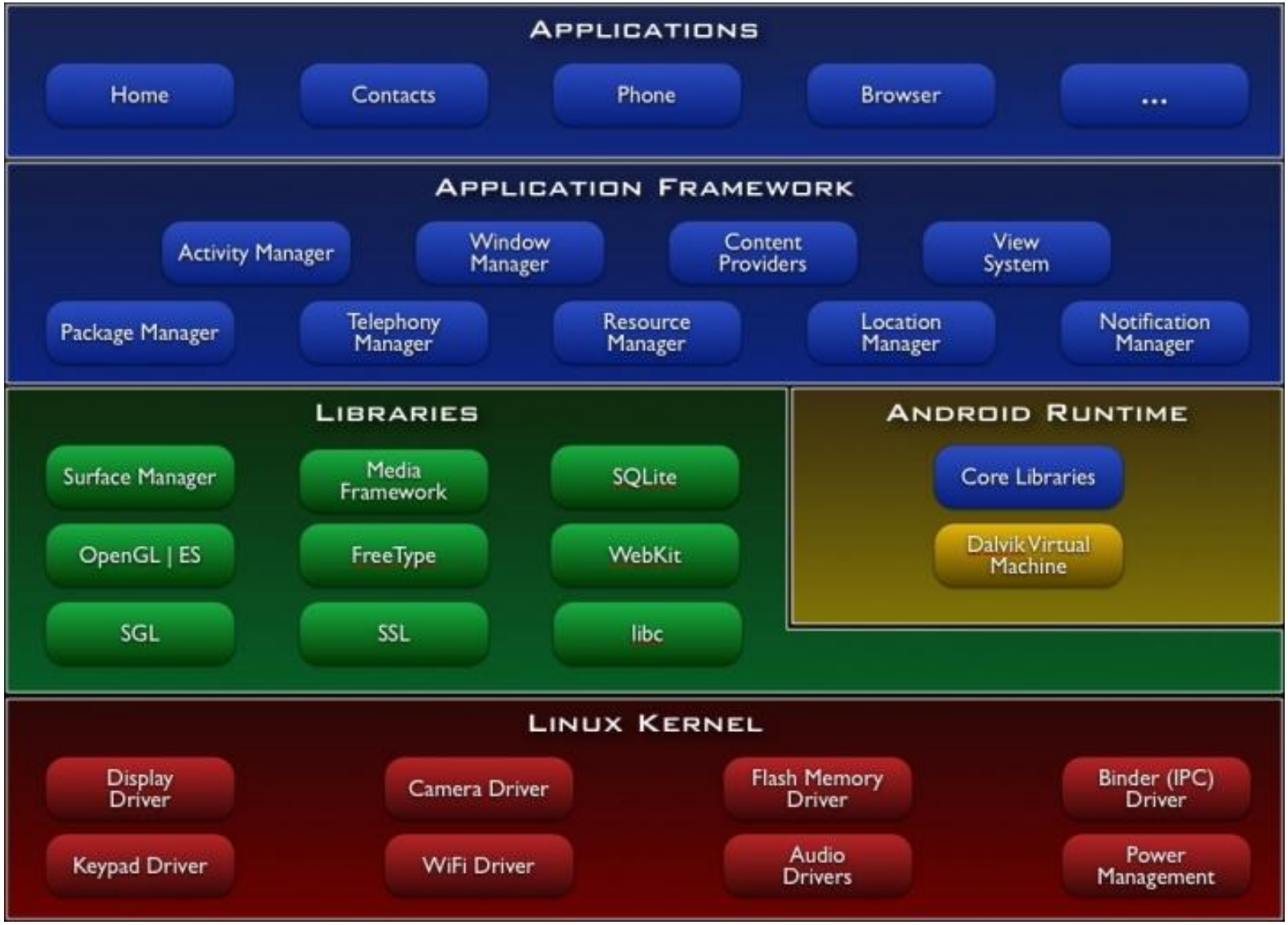
Open source operating system

- Google manages code and provides a market place

Developed by Open Handset Alliance (OHA) to run

- Smartphones
- Tablets
- Netbooks
- GoogleTV

Android Architecture



What does Android provide

Local storage

- SQL
- Key value pairs

Network connectivity

- Bluetooth, Wifi, Cell Network, SMS

Media

- Audio, video, image

Sensors

- Touch screen, accelerometer, gyroscope, compass, microphone

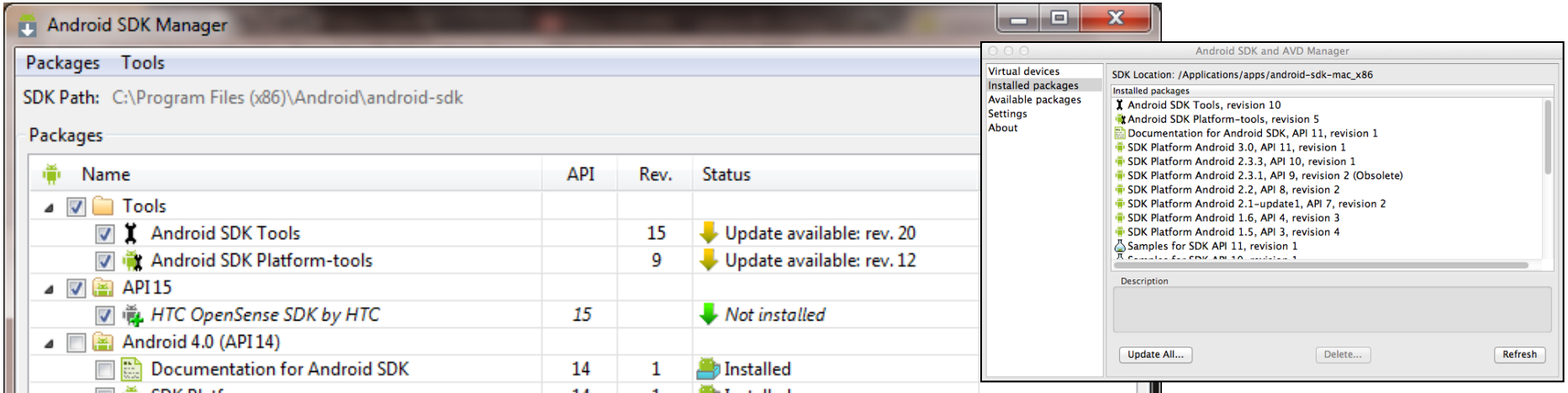
Tools

Free download - Software Developer Kit (SDK)

- http://developer.android.com/sdk/index.html
- Windows, Mac OS, Linux

Open Android SDK Tools + SDK Manager

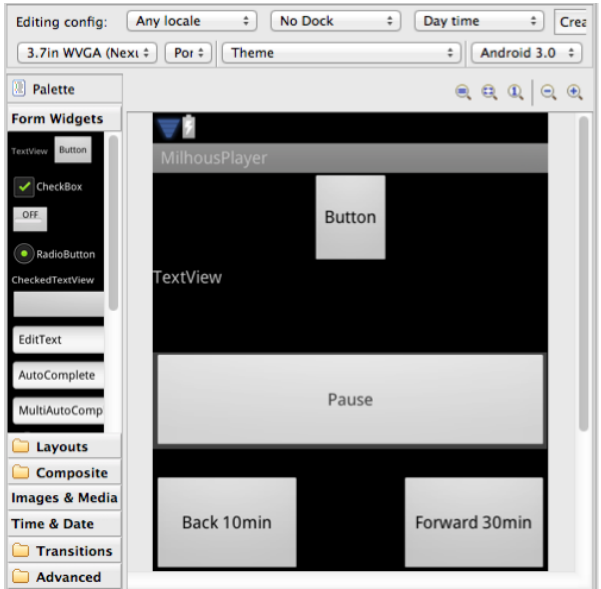
- /Applications/apps/android-sdk-mac_x86/tools/android
- Install the SDK for your target device
- Use AVD Manager to add virtual devices

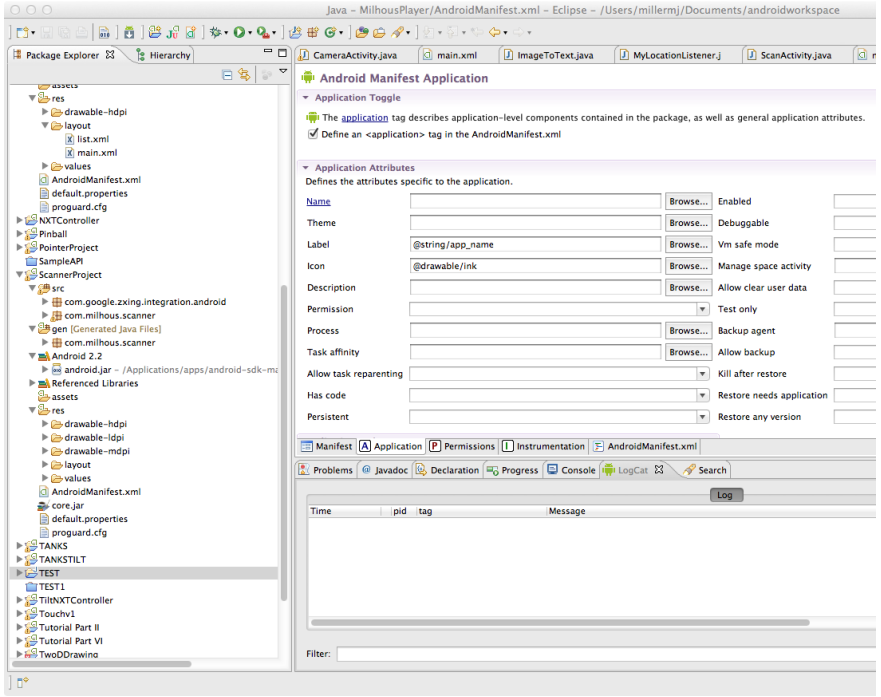


Tools

Plugin for the Eclipse IDE

- Android Development Tools (ADT)
- Provides integration between the SDK and devices
- WISIWYG layout editor
- Manage assets





Android Debug Bridge (adb)

Android Debug Bridge (adb)

- http://developer.android.com/guide/developing/tools/adb.html

Installed at

- C:\Program Files (x86)\Android\android-sdk\platform-tools
- /Applications/apps/android-sdk-mac_x86/platform-tools/

Connects to a device or emulator

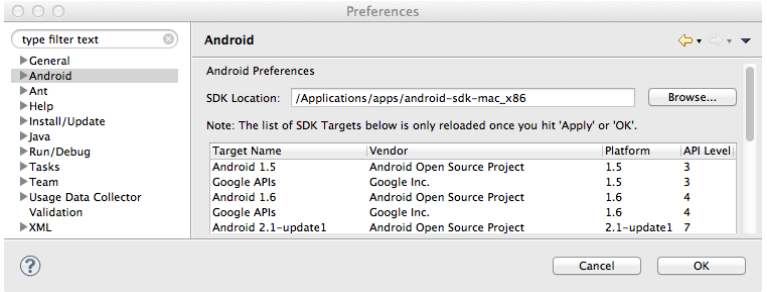
- Install applications, shell, push/pull files
- View Log messages

  - Filter to make manageable

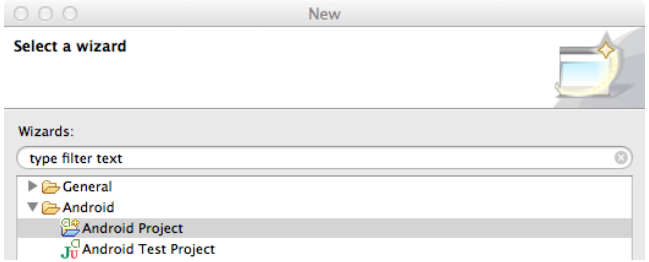
Create an application

Open eclipse

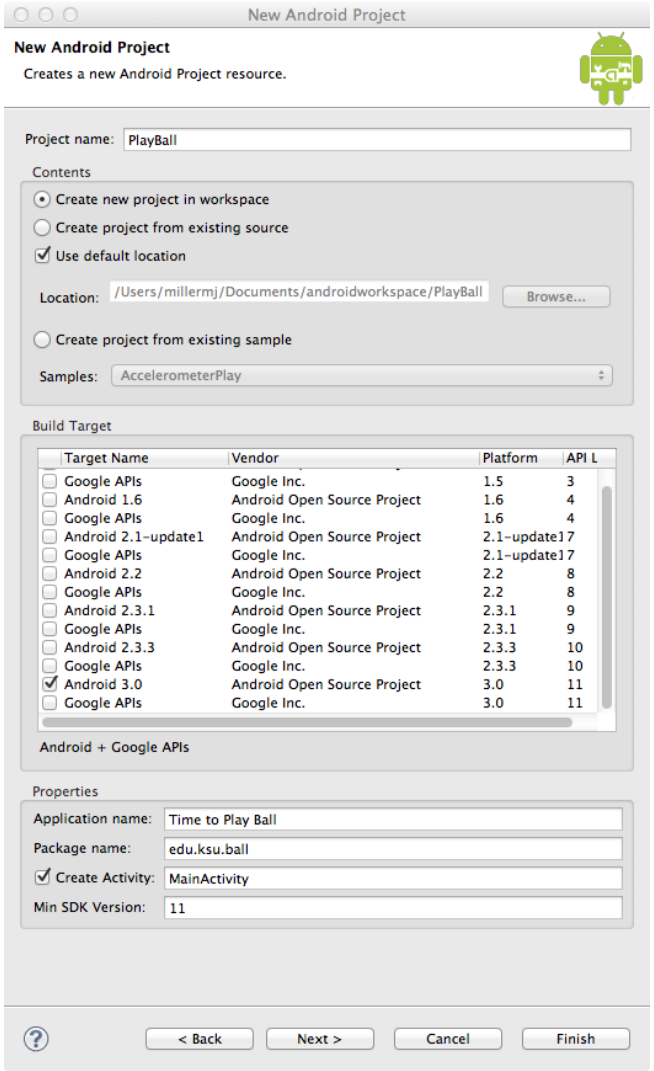
- Update settings to point to the platform-tools directory



Create new Project (other)



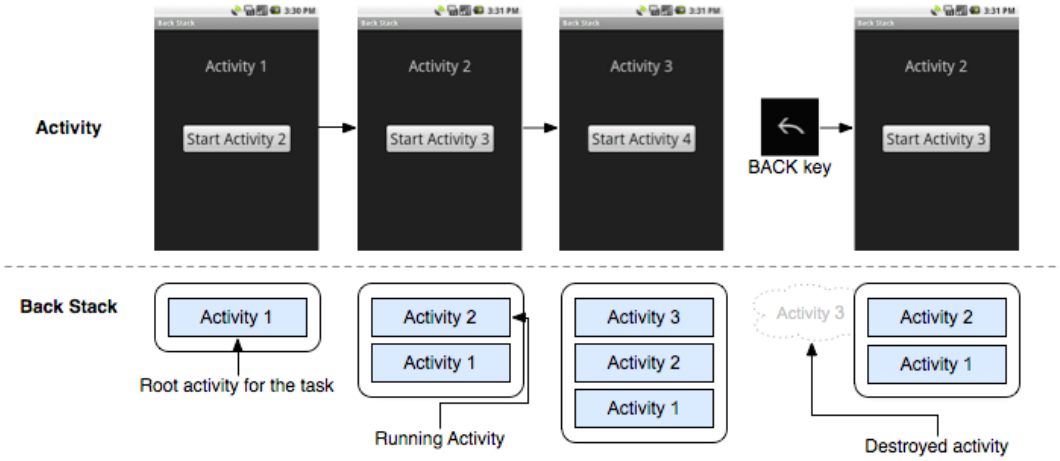
Fill in Application Information



Android Main concepts

Applications consist of Activities

- Allow users to interact with the Application
- User Interface
- Open other activities (stacked)



- Lifecycle

  Create, Stop, Resume, Destroy

Android Main concepts

Services

- Run in the background to do a task

Broadcast Receivers

- Receive system wide messages

  - Screen turned off, orientation change, picture captured

Intents

- Provide a way to send messages to Activities, Services and receive messages from the Broadcast Receivers

Created Artifacts

MainActivity.java

- The activity that is launched when the application starts

R.java

- Generated files for

  images, layouts and UI components

- Do not edit, created on build

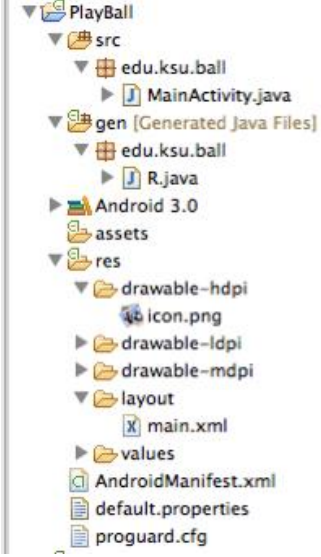
main.xml

- Starting layout for Main Activity

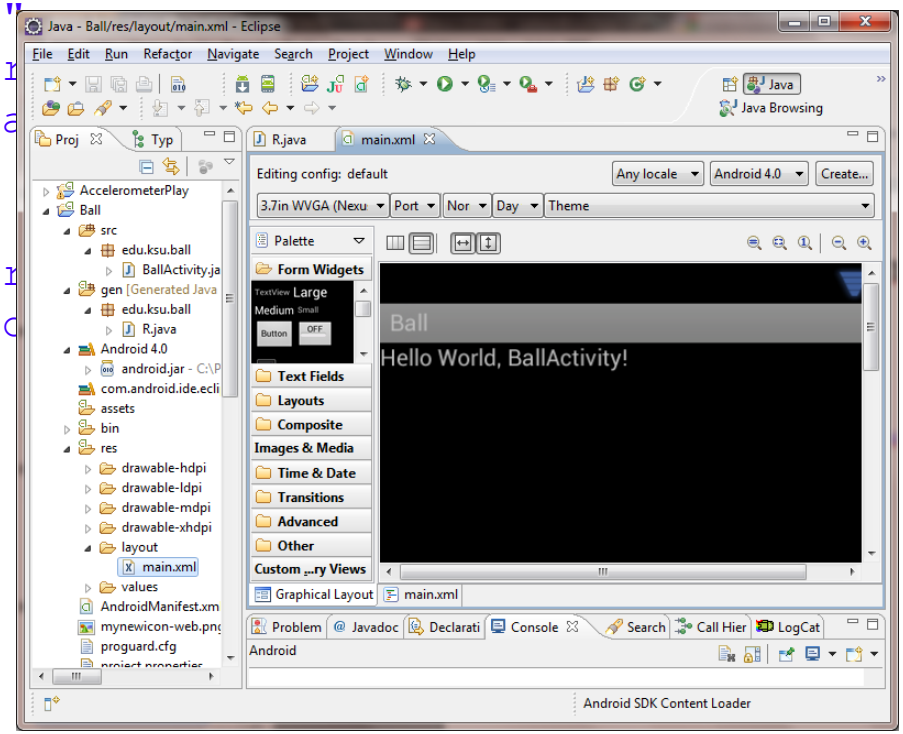
MainActivity.java

```
package edu.ksu.ball;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
        {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
        }
}
```

main.xml or activity_main.xml

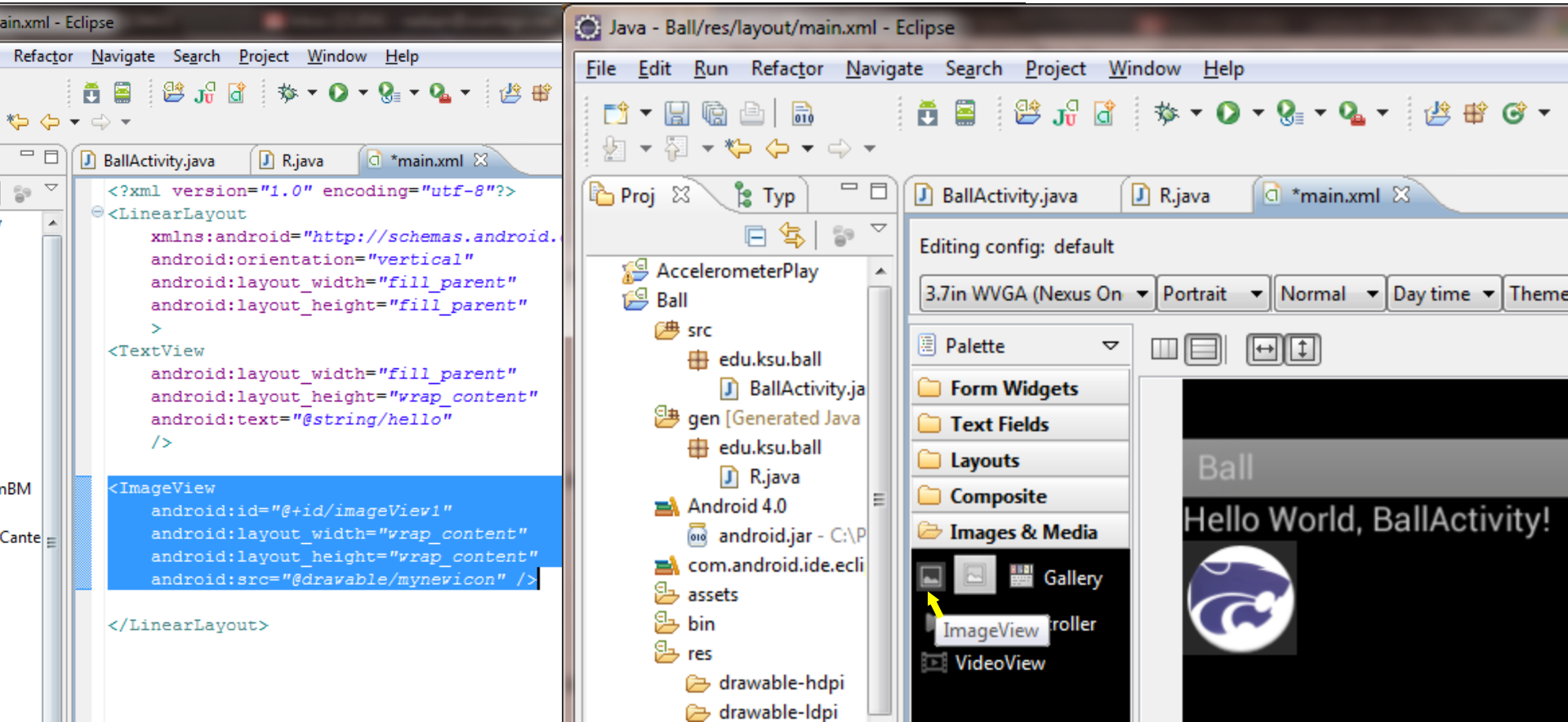



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_par
    android:layout_height="fill_pa
    >
<TextView
    android:layout_width="fill_par
    android:layout_height="wrap_co
    android:text="@string/hello"
    />
</LinearLayout>
```



Add an Image View

```
<ImageView
    android:id="@+id/imageView1"

android:layout_width="wrap_content"

android:layout_height="wrap_content
```



R.java - change is only visible after build

Java - Ball/gen/edu/ksu/ball/R.java - Eclipse

File Edit Run Source Refactor Navigate Search Project Window Help

Proj Typ

AccelerometerPlay
Ball
src
edu.ksu.ball
BallActivity.ja
gen [Generated Java
edu.ksu.ball
R.java
Android 4.0
android.jar - C:\P
com.android.ide.ecli
assets
bin
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
drawable-xhdpi
layout
main.xml
values
AndroidManifest.xm
mynewicon-web.png

R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.

package edu.ksu.ball;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
        public static final int mynewicon=0x7f020001;
    }
    public static final class id {
        public static final int imageView1=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

new id

Problem @ Javadoc Declarati Console Search Call Hier LogCat

Sensor Listener

Android provides asynchronous sensor framework

GPS, Accelerometer, Gyroscope, Bluetooth

Process for acquiring sensor data

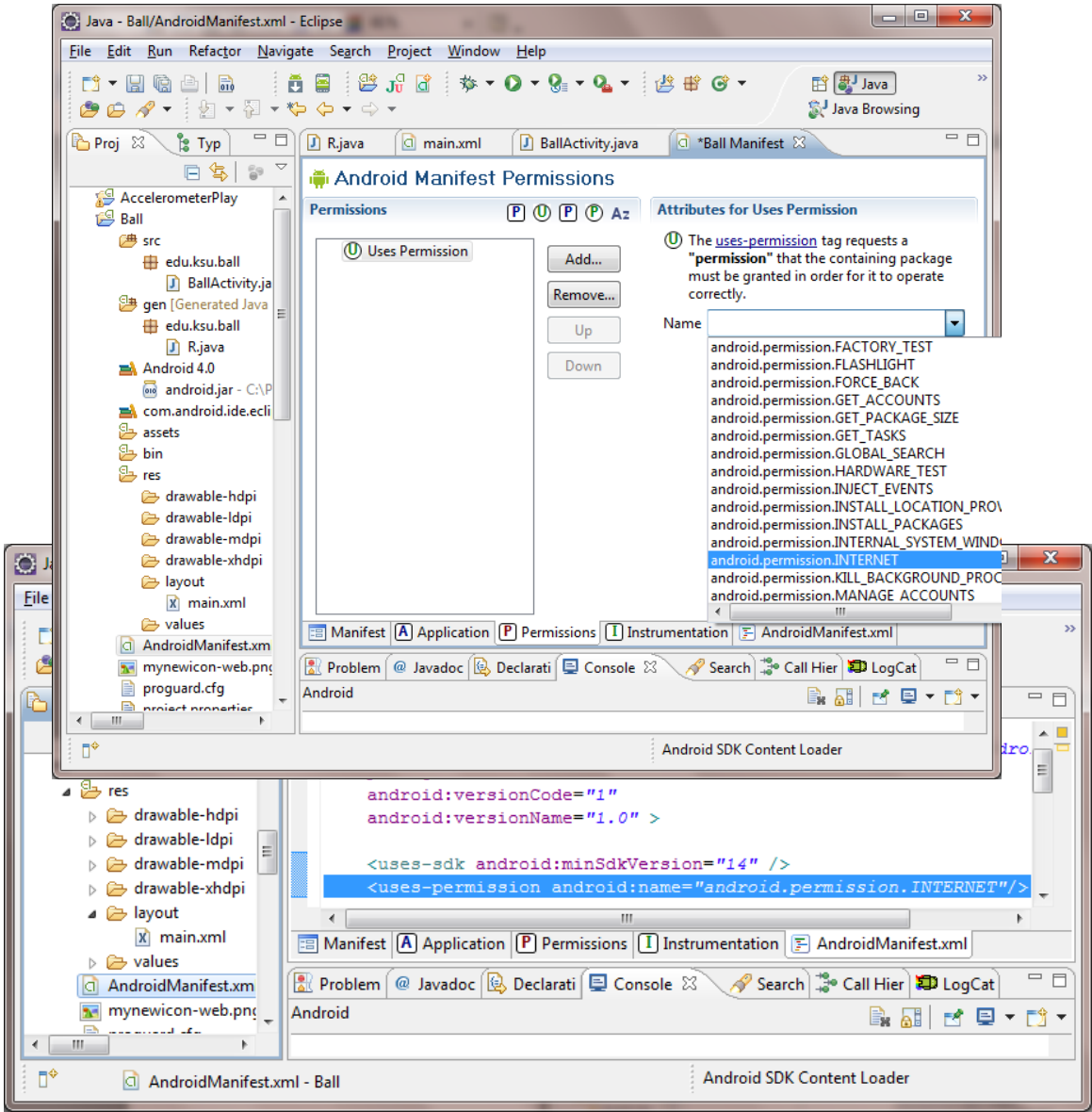
- Request permission in the app to listen to data
- Register a listener
- Process data

Requesting Permission

Required for

- Internet access
- Storage
- GPS
- Camera

Requests placed in AndroidManifiest.xml



Listener Example

Implement the SensorEventListener interface

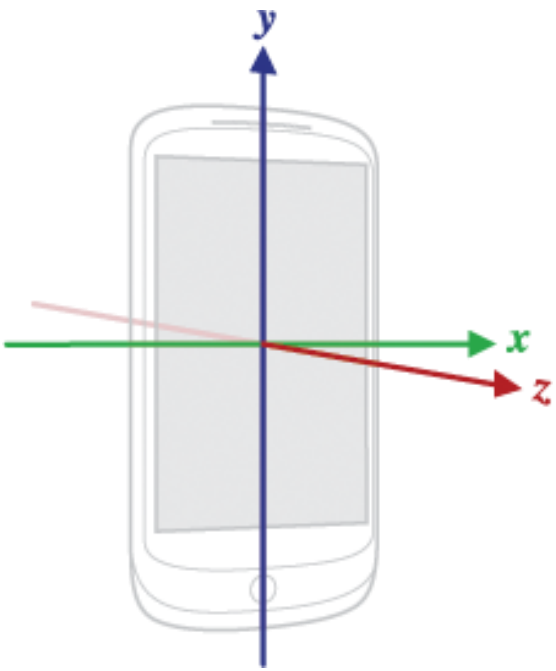
- public void onAccuracyChanged(Sensor sensor, int accuracy)
- public void onSensorChanged(SensorEvent event)

Register event listener

```
private SensorManager mSensorManager;
private Sensor mGyroScope;
...
mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
mGyroScope = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
```

Process data

RotationListener.java




```
public class RotationListener implements SensorEventListener
{
	private String rotation = "";

	private MainActivity main;

	public RotationListener(MainActivity main)
		{
			this.main = main;
		}

	@Override
	public void onAccuracyChanged(Sensor sensor, int accuracy)
		. . .
	@Override
	public void onSensorChanged(SensorEvent event)
		{
			switch (event.sensor.getType())
				{
				case Sensor.TYPE_GYROSCOPE:
					float[] values = event.values;
					rotation = String.format("x=%2.2f y=%2.2f z=%2.2f", values[0], values[1], values[2]);
					main.setText(rotation);
					break;
				}
		}
}
```

Buttons

Create button in User Interface builder

Add an onClickListener

```
Button myButton = (Button) findViewById(R.id.button1);
myButton.setOnClickListener(new View.OnClickListener()
{
          @Override
          public void onClick(View v)
               {
                    . . .
               }
});
```

Process Click

Intents

Provide a messaging between Activities and Services

Intents can provide specific user interface tailored to the intent

- Allows application to use other services and invoke other activities
- Examples

  Maps

  Barcode reader

  File selection

Intent Example

list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
    <ListView android:layout_height="600dip"
     android:layout_width="match_parent"
     android:id="@android:id/list">
</ListView>
</LinearLayout>
```

List id

Tell android about the activity

in AndroidMainfest.xml

```
<activity android:name="edu.ksu.ball.ListOfItems"  />
```

Intent Example (contd.)

ListOfItems.java

```
public class ListOfItems extends ListActivity
{
	public static final String SELECTED_ITEM = "selectedItem";

	String itemText = "";

	@Override
	public void onCreate(Bundle savedInstanceState)
		{
			super.onCreate(savedInstanceState);
			setContentView(R.layout.list);
			List<String> listOfItems = new LinkedList<String>();
			listOfItems.add("abc");
			listOfItems.add("def");
			listOfItems.add("xyz");
			this.setListAdapter(
			new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, listOfItems));
		}
```

Intent Example (contd.)

ListOfItems.java

```
    @Override
    protected void onListItemClick(ListView l, View v, int position, long id)
         {
              super.onListItemClick(l, v, position, id);
              itemText = getListView().getItemAtPosition(position).toString();
              finish();
         }

    @Override
    public void finish()
         {
              Intent data = new Intent();
              data.putExtra(SELECTED_ITEM, itemText);
              setResult(RESULT_OK, data);
              super.finish();
         }
}
```

Intent Example (contd.)

Create the new Intent MainActivity.java

```
public void onCreate(Bundle savedInstanceState)
     {
          . . .
          Button myButton = (Button) findViewById(R.id.mybutton);
          myButton.setOnClickListener(new View.OnClickListener()
          {

               @Override
               public void onClick(View v)
                    {
                         Intent i = new Intent(MainActivity.this, ListOfItems.class);
                         startActivityForResult(i, REQUEST_CODE);
                    }
          });

          . . . .

protected void onActivityResult(int requestCode, int resultCode, Intent data)
     {
          if (resultCode == RESULT_OK && requestCode == REQUEST_CODE)
               {
                    if (data.hasExtra(ListOfItems.SELECTED_ITEM))
                         {
                              String text = data.getExtras().getString(ListOfItems.SELECTED_ITEM);
                              Log.d("PlayBall", text);
                         }
               }
     }
```

Services

Allow an application to execute a long running task

Does not require user input

Two types

- Bound services

  Allow two way communication between application and the service

- Unbound services

  No implicit two way communication

Examples

- http://developer.android.com/guide/topics/fundamentals/services.html

Service Example

```
public class BoundBallService extends Service
{

     private final IBinder myBinder = new BallBinder(this);

     private Looper mServiceLooper;

     private ServiceHandler mServiceHandler;

     @Override
     public void onCreate()
           {
                 // Start up the thread running the service. Note that we create a
                 // separate thread because the service normally runs in the process's
                 // main thread, which we don't want to block. We also make it
                 // background priority so CPU-intensive work will not disrupt our UI.
                 HandlerThread thread = new
                 HandlerThread("ServiceStartArguments", android.os.Process.THREAD_PRIORITY_BACKGROUND);
                 thread.start();

                 // Get the HandlerThread's Looper and use it for our Handler
                 mServiceLooper = thread.getLooper();
                 mServiceHandler = new ServiceHandler(this, mServiceLooper);
           }
```

Service Example

```
    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
          {
                Log.d("PlayBall", "Service start");
                Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show();

                // For each start request, send a message to start a job and deliver the
                // start ID so we know which request we're stopping when we finish the job
                Message msg = mServiceHandler.obtainMessage();
                msg.arg1 = startId;
                mServiceHandler.sendMessage(msg);

                // If we get killed, after returning from here, restart
                return START_STICKY;
          }

    @Override
    public IBinder onBind(Intent intent)
          {
                return myBinder;
          }

    @Override
    public void onDestroy()
          {
                Toast.makeText(this, "service done", Toast.LENGTH_SHORT).show();
          }

}
```

Service Handler

```
final class ServiceHandler extends Handler
{

	private final BoundBallService BallServiceHandler;

	public ServiceHandler(BoundBallService boundBallService, Looper looper)
		{
			super(looper);
			BallServiceHandler = boundBallService;
		}

	@Override
	public void handleMessage(Message msg)
		{
			// Normally we would do some work here, like download a file.
			// For our sample, we just sleep for 5 seconds.
			long endTime = System.currentTimeMillis() + 5 * 1000;
			while (System.currentTimeMillis() < endTime)
				{
					synchronized (this)
						{
							try
								{
								wait(endTime - System.currentTimeMillis());
								}
							catch (Exception e)
								{
								}
						}
				}
			// Stop the service using the startId, so that we don't stop
			// the service in the middle of handling another job
			BallServiceHandler.stopSelf(msg.arg1);
		}
}
```

Using Bound Service

Create an Intent to initiate the service

Call the bindService method

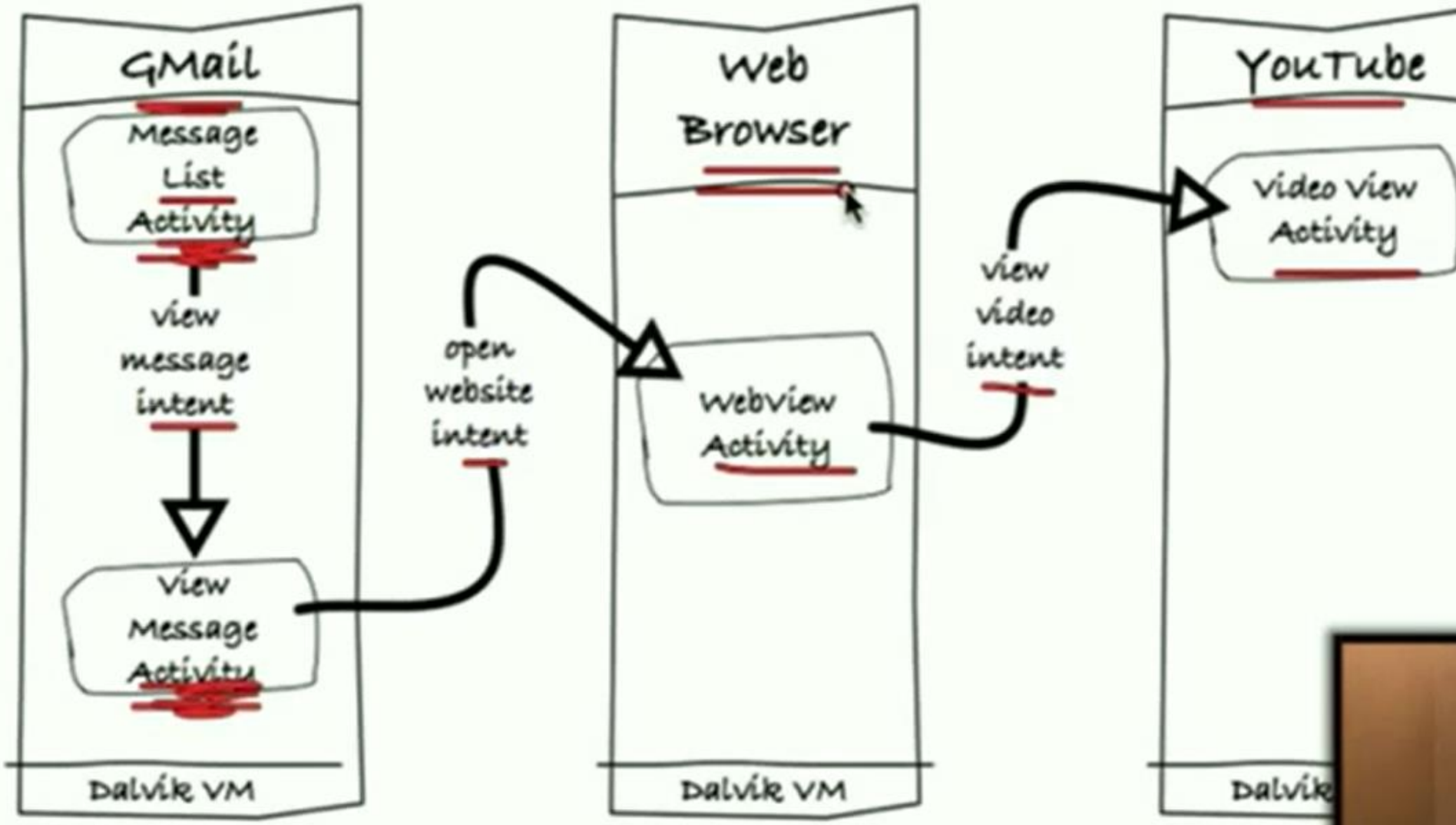
```
BallServiceConnection connection = new BallServiceConnection(this);
Intent intent = new Intent(this, BoundBallService.class);
bindService(intent, connection, Context.BIND_AUTO_CREATE);
```

Wait until the service has been created

Use methods defined in the service

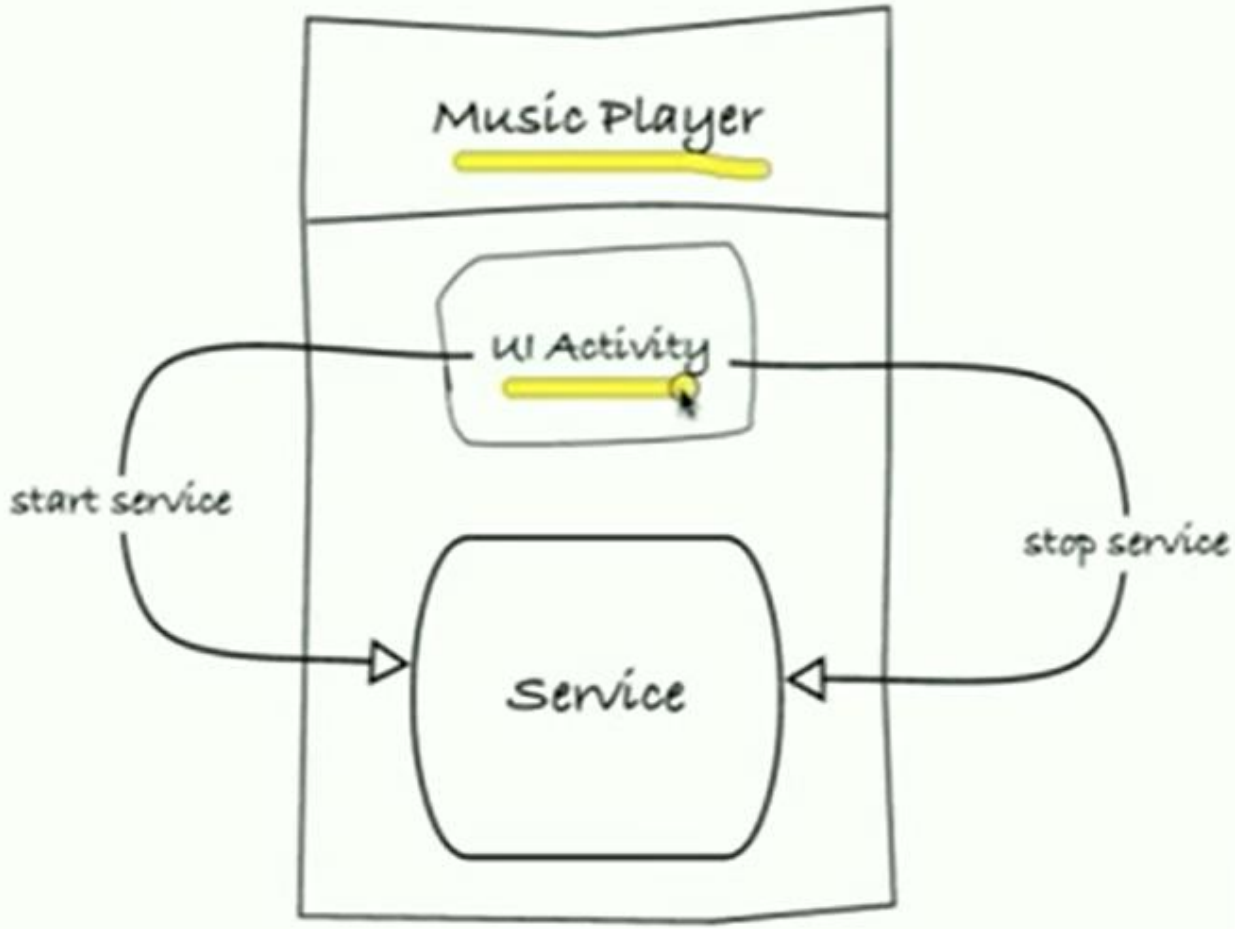
Overview

http://www.youtube.com/watch?v=h3gPo7qFOFw



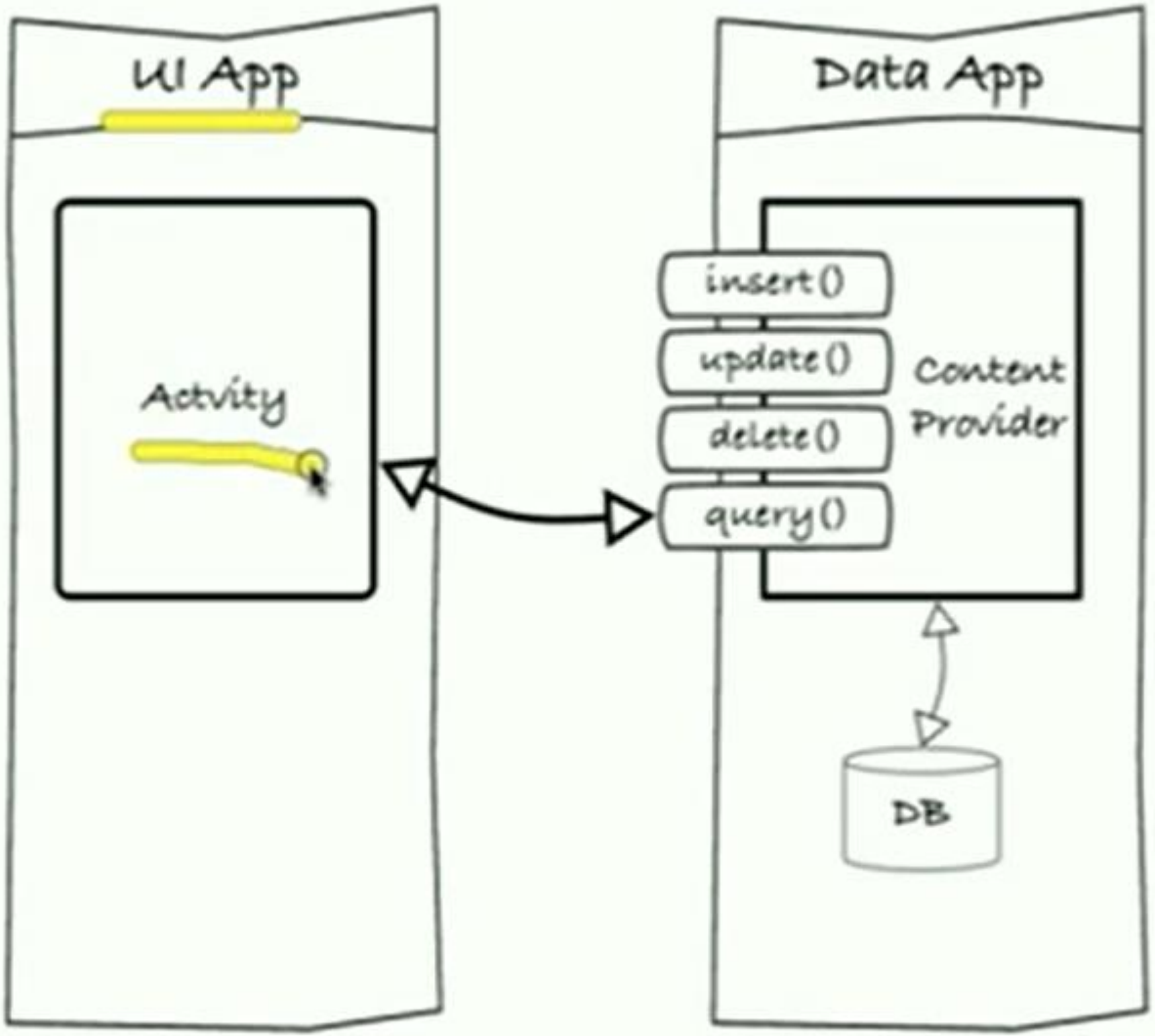
are like events or messages.

Service Overview

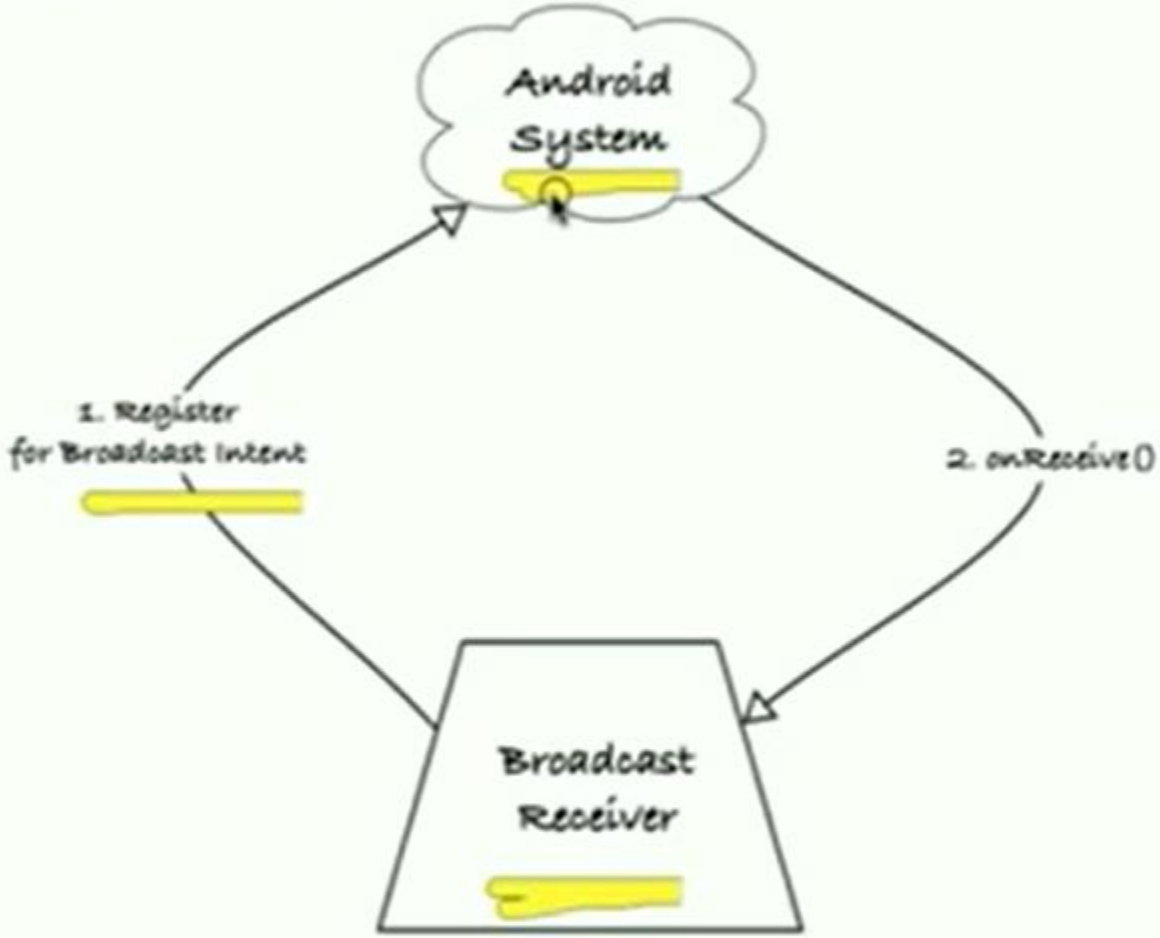


- Services are code that runs in the background.
- They can be started and stopped. Services doesn't have UI.
- Keep in mind that service runs on the main application thread, the UI thread.

Typical Usage of Content Providers



Broadcast Receiver Overview



- An Intent-based publish-subscribe mechanism.
- Great for listening system events such as SMS messages.

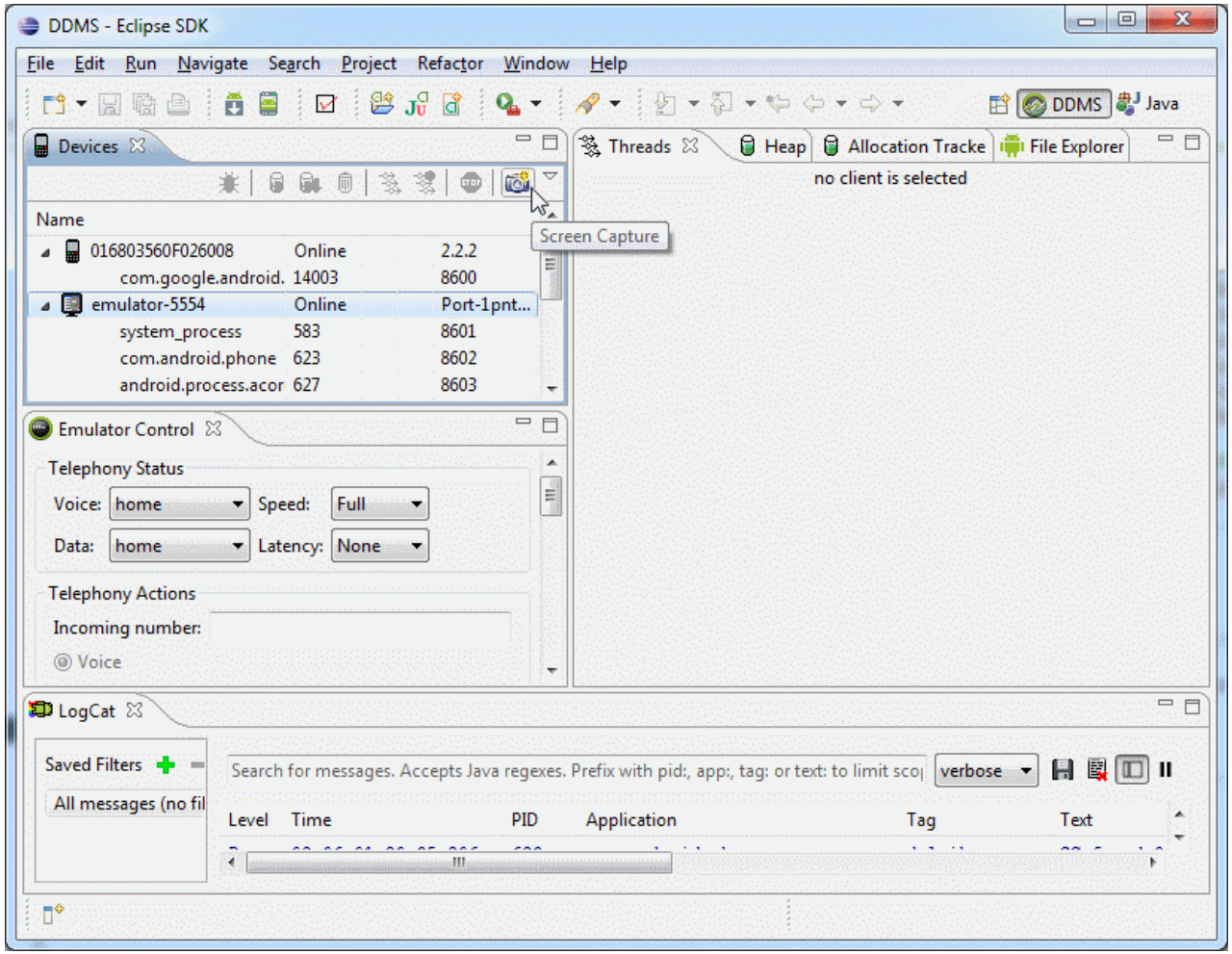
DDMS

- Debugging tool called the Dalvik Debug Monitor Server (DDMS)
- Run From Eclipse: Click Window > Open Perspective > Other... > DDMS

DDMS features

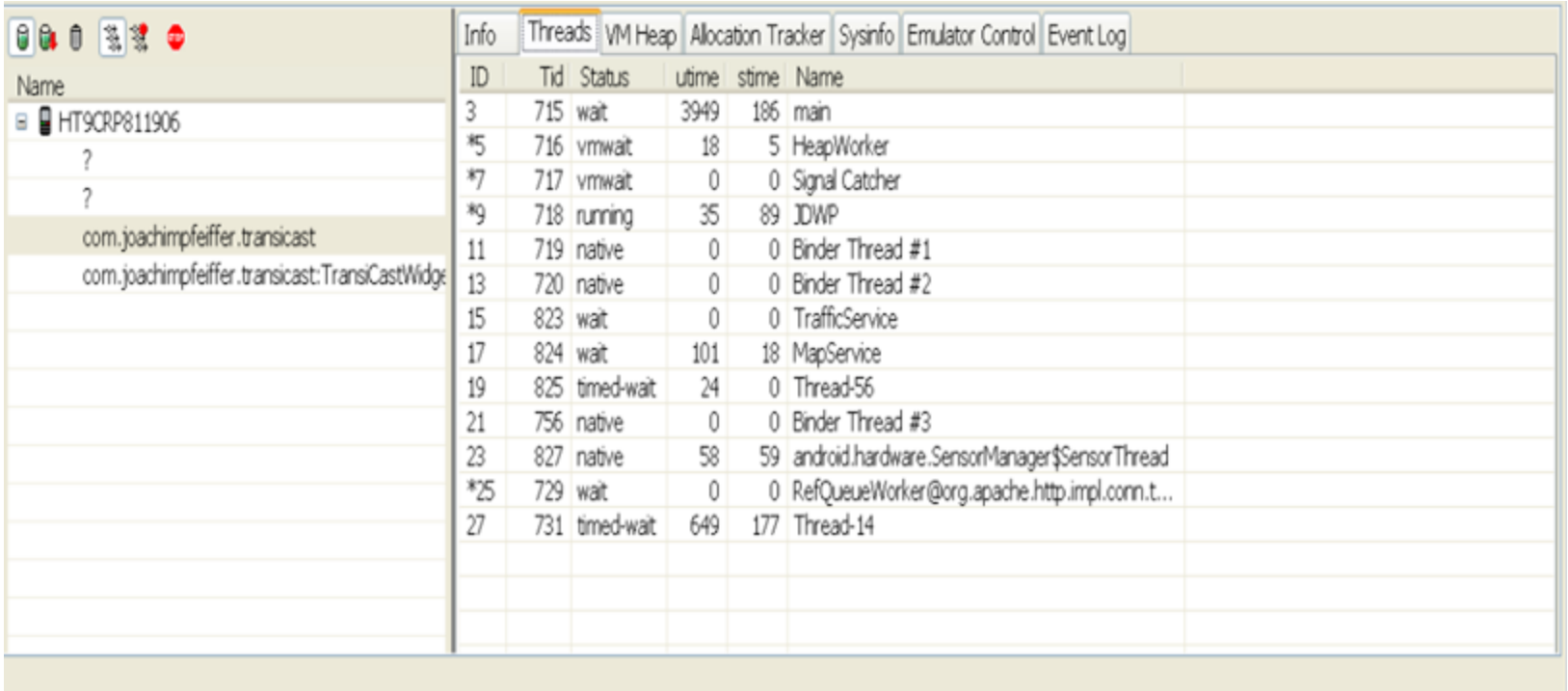
- Screen capture
- Thread and heap information
- Tracking memory allocation of objects
- Logcat
- System info
- Call and location data spoofing

DDMS features



DDMS features

Thread information



ID	Tid	Status	utime	stime	Name
3	715	wait	3949	186	main
*5	716	vmwait	18	5	HeapWorker
*7	717	vmwait	0	0	Signal Catcher
*9	718	running	35	89	JDWP
11	719	native	0	0	Binder Thread #1
13	720	native	0	0	Binder Thread #2
15	823	wait	0	0	TrafficService
17	824	wait	101	18	MapService
19	825	timed-wait	24	0	Thread-56
21	756	native	0	0	Binder Thread #3
23	827	native	58	59	android.hardware.SensorManager$SensorThread
*25	729	wait	0	0	RefQueueWorker@org.apache.http.impl.conn.t...
27	731	timed-wait	649	177	Thread-14

DDMS features

Heap information

Dalvik Debug Monitor

File Edit Actions Device Help

Name

- HT9CRP811906
  - ?
  - ?
  - com.joachimpfeiffer.transicast
  - com.joachimpfeiffer.transicast:TransiCastWidge

Info | Threads | VM Heap | Allocation Tracker | Sysinfo | Emulator Control | Event Log

Heap updates will happen after eve

ID	Heap Size	Allocated	Free	% Used	# Objects
1	5.695 MB	3.719 MB	1.977 MB	65.29%	59,265

Cause GC

Display: Stats

Type	Count	Total Size	Smallest	Largest	Median	Average
free	2,642	1.946 MB	16 B	237.148 KB	40 B	772 B
data object	37,879	1.183 MB	16 B	808 B	32 B	32 B
class object	3,213	552.992 KB	176 B	184 B	176 B	176 B
1-byte array (byte[], boolean[])	1,123	884.266 KB	24 B	6.516 KB	56 B	806 B
2-byte array (short[], char[])	13,624	864.500 KB	24 B	28.023 KB	48 B	64 B
4-byte array (object[], int[], float[])	3,394	289.195 KB	24 B	16.023 KB	48 B	87 B
8-byte array (long[], double[])	32	4.914 KB	32 B	1.000 KB	104 B	157 B
non-Java object	345	31.289 KB	16 B	8.023 KB	32 B	92 B

DDMS features

Tracking memory allocation of objects

Dalvik Debug Monitor

Name		
HT833GZ01048	Online	
system_process	61	
com.android.phone	188	
android.process.acore	192	
com.google.process.gapps	235	
com.android.alarmclock	238	
com.google.android.apps.maps	329	
com.google.android.gm	357	
org.curiouscreature.android.shelves	916	
com.google.zxing.client.android	931	

Info | Threads | VM Heap | Allocation Tracker | Sysinfo | Emulator Control | Event Log

Start Tracking | Get Allocations

...	Allocated Class	Thread Id	Allocated in	Allocated in
244	char[]	3	java.lang.String	<init>
200	byte[]	9	org.apache.harmony.dalvik.ddmc.DdmVmInternal	getThreadStats
134	char[]	3	java.lang.String	<init>
122	char[]	3	java.lang.String	<init>
104	char[]	3	java.lang.String	<init>
102	char[]	3	java.lang.String	<init>
96	char[]	3	java.lang.String	<init>
84	char[]	3	java.lang.String	<init>
80	char[]	3	java.lang.String	<init>
80	char[]	3	java.lang.String	<init>
76	char[]	3	java.lang.String	<init>
66	char[]	3	java.lang.String	<init>
66	char[]	3	java.lang.String	<init>

Class	Method	File	Line	Native
java.lang.String	<init>	String.java	512	false
android.text.TextUtils	blank	TextUtils.java	1249	false
android.text.TextUtils	ellipsize	TextUtils.java	1051	false
android.text.BoringLayout	replaceOrMake	BoringLayout.java	104	false
android.widget.TextView	makeNewLayout	TextView.java	4266	false
android.widget.TextView	checkForRelayout	TextView.java	4688	false
android.widget.TextView	setText	TextView.java	2523	false
android.widget.TextView	setText	TextView.java	2393	false

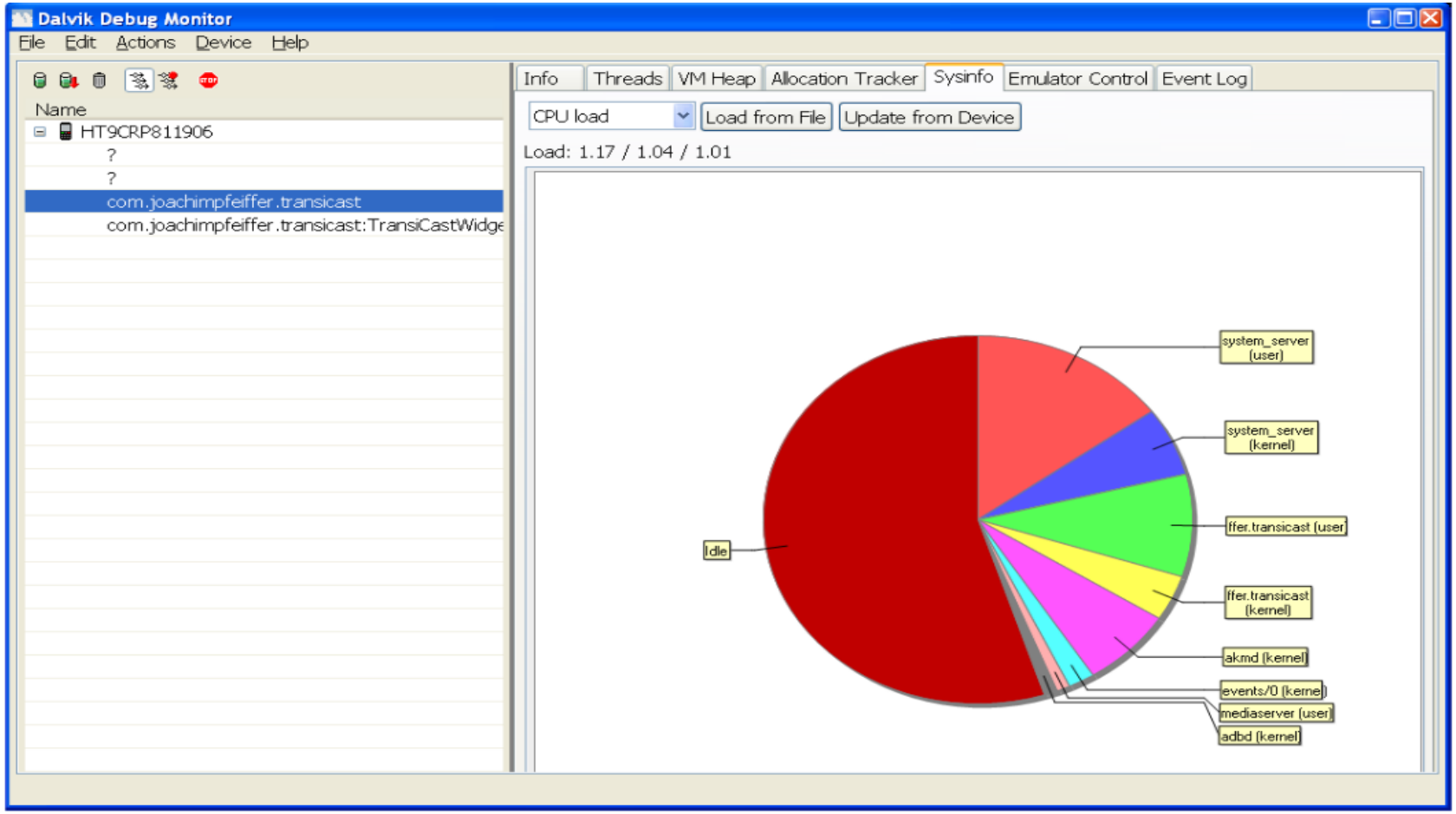
DDMS features

Logcat

- Mechanism for collecting and viewing system debug output
- Logcat dumps a log of system messages, stack traces when error occurs and user messages written with the Log class
- You can filter the log for viewing only messages from a particular process, error messages, debug messages, warning messages, etc.

DDMS features

System info



Android Monkey Runners

- The Monkey is a program that runs on your emulator or device and generates random events such as clicks, touches, or gestures, as well as a number of system-level events.

- You can use it for testing your application

- You can set a number of options like number of event to attempt, event types and frequencies, delay between the events

- **How to run monkey?**

  > **adb shell monkey -v -p your.package.name 300**

Android Debug Bridge (ADB)

What is ADB?

- Command-line tool that lets you communicate with an emulator instance or connected Android-powered device

Contains three major components

- Client – on development machine
- Server – background process on development machine
- Daemon – on emulator or device

Android Debug Bridge (ADB)

What it does?

- Issue shell commands
- Push files to the device
- Pull files from device
- Install files remotely

Android Debug Bridge (ADB)

How to execute a command?

- adb -s <serialNumber> <command>

Major ADB Commands:

- adb devices - List of devices attached
- adb install <path_to_apk>
- adb push foo.txt /sdcard/foo.txt

Android Debug Bridge (ADB)

Major ADB Commands:

- adb pull /sdcard/foo.txt foo.txt
- adb shell
- adb shell [<shellCommand>]

References..

- http://developer.android.com/guide/index.html
- http://developer.android.com/guide/topics/manifest/uses-library-element.html
- http://developer.android.com/guide/developing/tools/adb.html
- http://developer.android.com/guide/developing/debugging/ddms.html
- http://developer.android.com/sdk/eclipse-adt.html