ActionBar, SideSwipe and Navigation Drawer

Kari Salo
Principal lecturer, Mobile Media
Metropolia University of Applied Sciences

Metropolia

ToC

- Action Bar
  - Action Buttons
  - Navigation Tabs
  - Split Action Bar
- Side Swipe
- Navigation Drawer
- Contextual Action Mode – self study

Up and Back Navigation

- Read and understand these principles:
  - http://developer.android.com/design/patterns/navigation.html
  - http://developer.android.com/training/implementing-navigation/ancestral.html
- Android 1.x - 2.3.x: only Back button

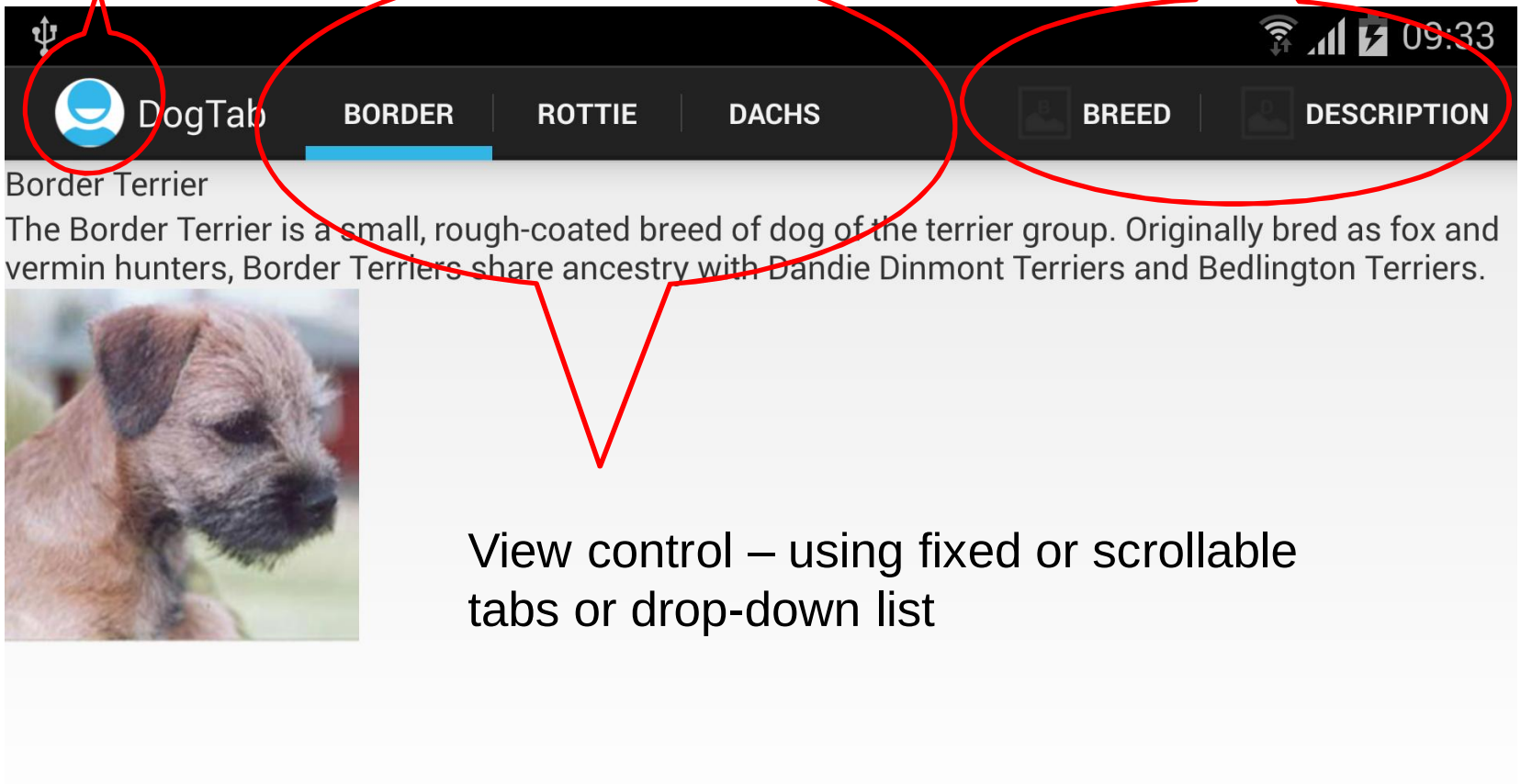
- Android 3.x - : Back and Up button

- Android 4.1 - : android:parentActivityName attribute + NavUtils APIs (support v4 package)
- The Up button is used to navigate within an app based on the hierarchical relationships between screens
- The system Back button is used to navigate, in reverse chronological order, through the history of screens the user has recently worked with.

ActionBar

App icon
(+optionally Up
navigation)

Action
buttons



View control – using fixed or scrollable
tabs or drop-down list

Action Bar with Action Buttons

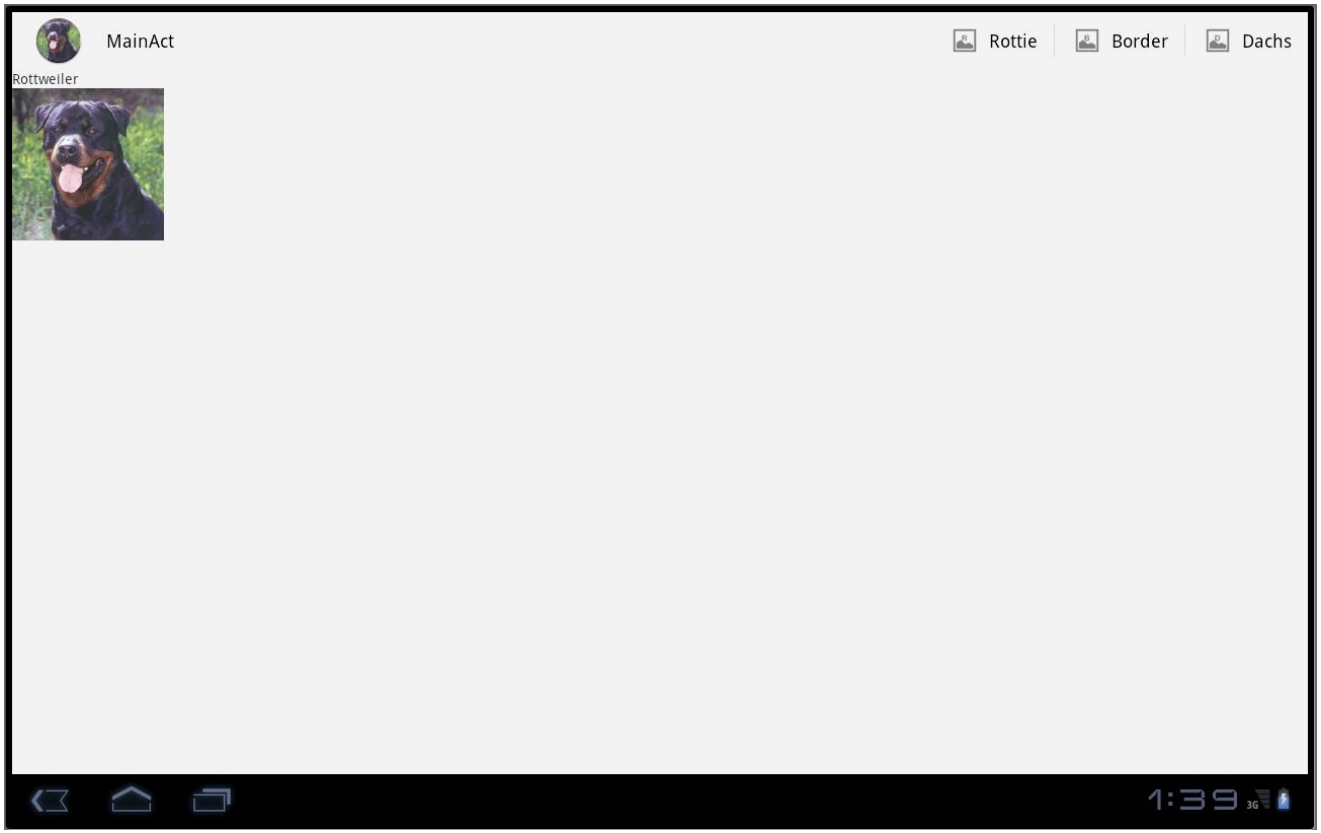
- Action bar is included in all activities that use the Theme.Holo or its descendants
  - *android:Theme.Holo.Light*
- Options menu implements Action buttons
  - Define menu items in xml-file
  - Implement in Activity and/or in Fragment:

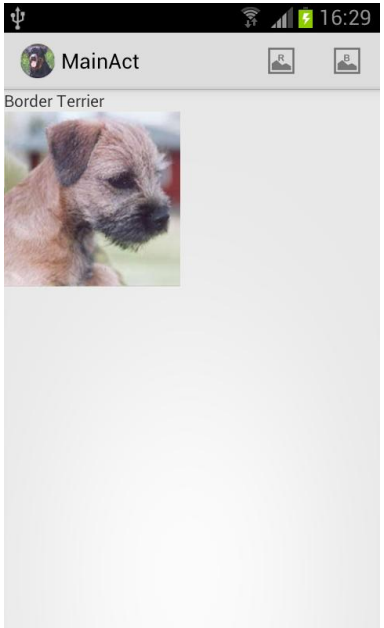
```
public boolean onCreateOptionsMenu(Menu menu) {
        ...
}
```

  - Implement

```
public boolean onOptionsItemSelected(MenuItem item) {
        ...
}
```

Dog Example


MainAct
Rottie
Border
Dachs
Rottweiler
1:39


16:29
MainAct
Border Terrier

Example

```
public class MainAct extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.mymenu, menu);
        return true;
    }
```

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/rottie"
        android:icon="@drawable/ic_action_rottie"
        android:title="@string/menu_r"
        android:showAsAction="ifRoom|withText" />
<item android:id="@+id/border"
        android:icon="@drawable/ic_action_border"
        android:title="@string/menu_b"
        android:showAsAction="ifRoom|withText" />
<item android:id="@+id/dachs"
        android:icon="@drawable/ic_action_dachs"
        android:title="@string/menu_d"
        android:showAsAction="ifRoom|withText" />
</menu>
```

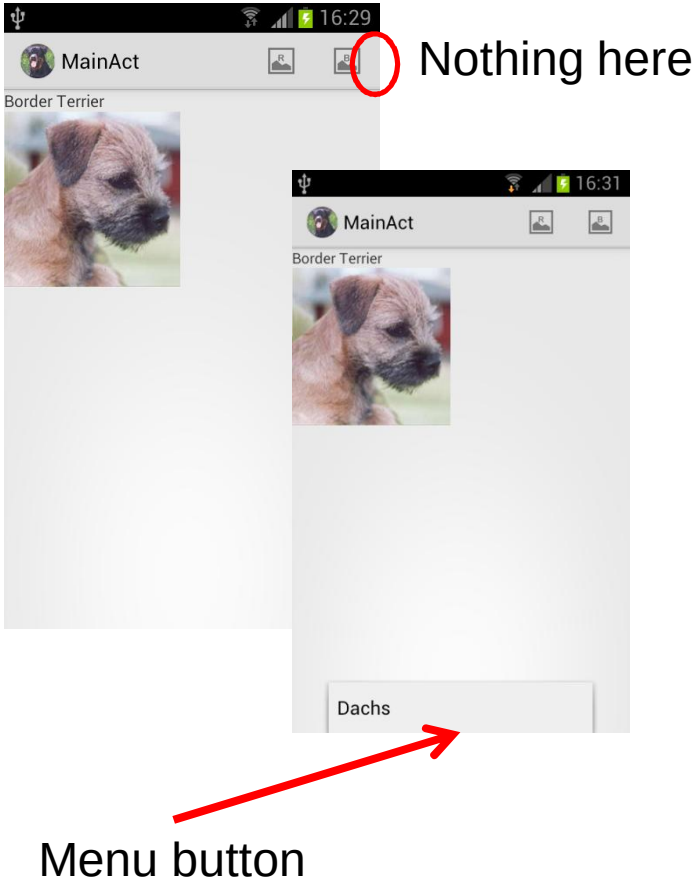
Example

```
public boolean onOptionsItemSelected(MenuItem item){
        TextView tv = (TextView)findViewById(R.id.Tv1);
        ImageView im = (ImageView)findViewById(R.id.Img1);
        switch(item.getItemId()) {
         case R.id.border:
                tv.setText("Border Terrier");
                im.setImageResource(R.drawable.borderterrier);
                break;
         case R.id.rottie:
                tv.setText("Rottweiler");
                im.setImageResource(R.drawable.rottweiler);
                break;
         case R.id.dachs:
                tv.setText("Dachshund");
                im.setImageResource(R.drawable.dachshund);
                break;
        }
        return true;   } }
```

Overflow Menu

- User friendly ?

The overflow menu is revealed either by the device *Menu* button (if provided by the device) or an additional button in the action bar (if the device does not provide the *Menu* button).



Logo or App Icon

- If android:logo attribute is defined then the action bar uses the logo image instead of the icon.

```
<application
    android:icon="@drawable/ic_launcher"
    android:logo="@drawable/metropolia"
    …
```

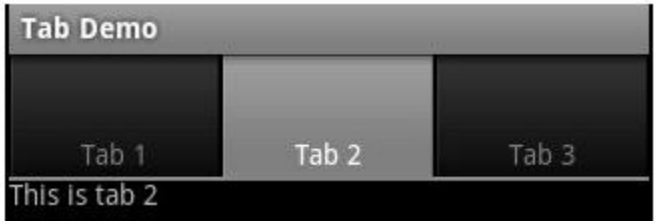
- App Icon/Logo for navigation:
  - When the user touches the icon/logo, the system calls activity's onOptionsItemSelected() method with the android.R.id.home
  - Android 4.x you should add:
    - ActionBar actionBar = getActionBar();
      actionBar.setHomeButtonEnabled(true);

    or

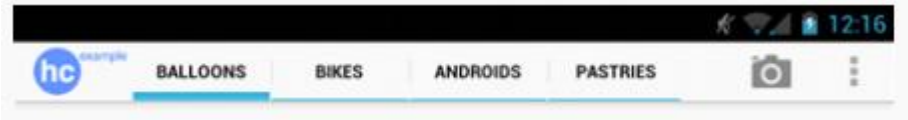
    - ActionBar actionBar = getActionBar();
      actionBar.setDisplayHomeAsUpEnabled(true);

Tab Navigation

- Old way: You will see a lot of Tab Navigation tutorials that use TabSpec and TabActivity (deprecated) classes, and layout consisting of TabHost/TabWidget/FrameLayout



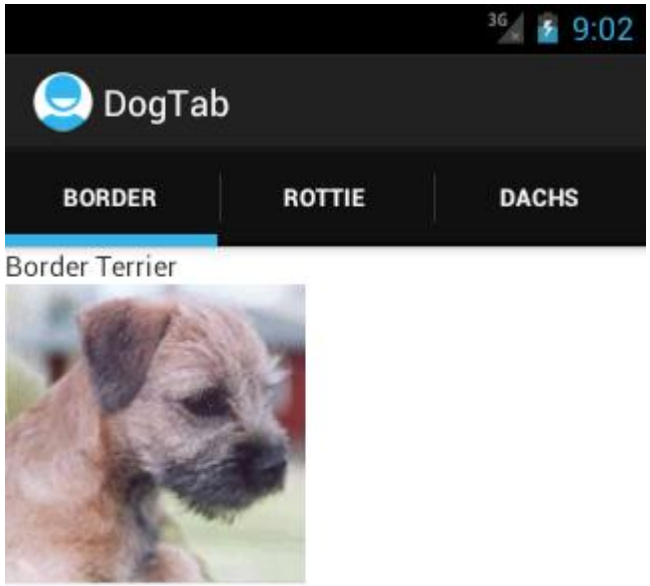
- Recommended way: The action bar provides built-in tab navigation for switching between fragments.



Action Bar with Navigation Tabs

- Basic idea: **switch between fragments using the tabs**
- Layout must include a ViewGroup in which you place each Fragment associated with a tab
- Define Fragments and their layout
- Implement the ActionBar.TabListener interface. Callbacks in this interface respond to user events on the tabs so you can swap fragments.
- For each tab you want to add, instantiate an ActionBar.Tab and set the ActionBar.TabListener by calling setTabListener(). Also set the tab's title and/or icon with setText() and/or setIcon().
- Add each tab to the action bar by calling addTab().

Dog Example



Activity and Layout

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ActionBar actionBar = getActionBar();
    actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
    Fragment TF1 = new Tab1Fragment();
    Fragment TF2 = new Tab2Fragment();
    Fragment TF3 = new Tab3Fragment();

    String label1 = getResources().getString(R.string.label1);
    Tab tab1 = actionBar.newTab();
    tab1.setText(label1);
    TListener t1 = new TListener(TF1);
    tab1.setTabListener(t1);
    actionBar.addTab(tab1);
...
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schema..."
    android:id="@+id/frag_container"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</LinearLayout>
```

Fragment and Layout

```
public class Tab1Fragment extends Fragment {

    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedState) {
        LinearLayout myView =(LinearLayout) inflater.inflate(R.layout.tab, container, false);
        TextView tv = (TextView) myView.findViewById(R.id.Tv1);
        ImageView im = (ImageView) myView.findViewById(R.id.Img1);
        tv.setText("Border Terrier");
        im.setImageResource(R.drawable.borderterrier);
        return myView;
    }

}
```

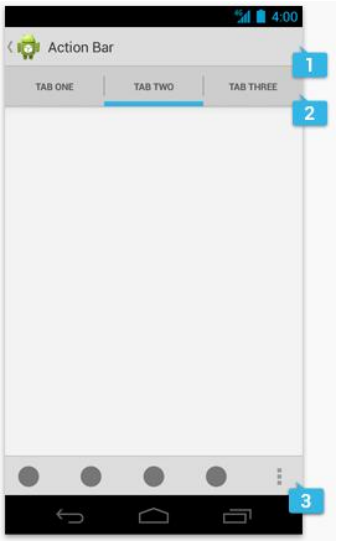
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/Tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <ImageView
        android:id="@+id/Img1"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:scaleType="fitXY" />
</LinearLayout>
```

Note: Fragment does not support findViewById()

TabListener Interface

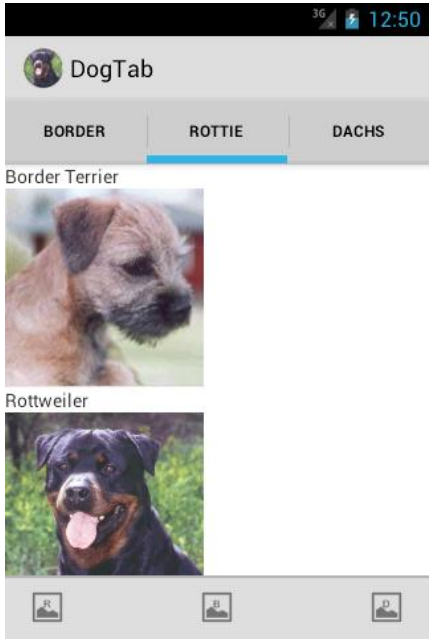
```
public class TListener implements ActionBar.TabListener {
        private Fragment mFragment;

        public TListener(Fragment frag) {
                mFragment = frag;
        }

        public void onTabSelected(Tab tab, FragmentTransaction ft) {
                ft.replace(R.id.frag_container, mFragment);
        }

        public void onTabUnselected(Tab tab, FragmentTransaction ft) {
        }

        public void onTabReselected(Tab tab, FragmentTransaction ft) {
        }
}
```

Split ActionBar



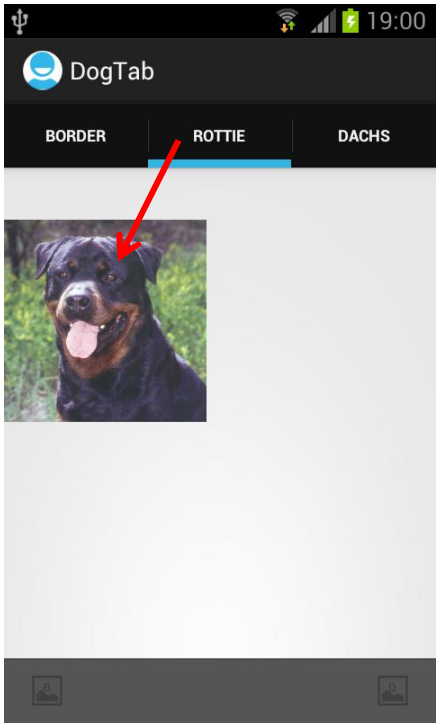
1. Main action bar (App Icon)
2. Top bar (Tabs)
3. Bottom bar (Action Buttons)

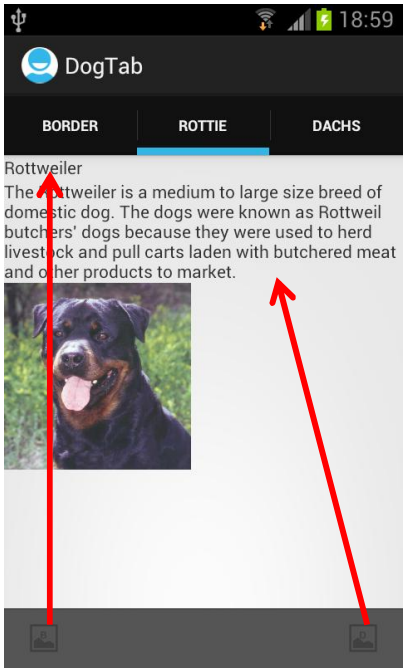
To implement Split Actions you only need to add one attribute in manifest:
android:uiOptions="*splitActionBarWhenNarrow*"



Lab 4

- Implement ”Dog Show”
- **Split Action Bar**
- **Dog image can be selected from Tabs**
- **Action Buttons Display**
  - **Dog breed in textual format**
  - **Short description of the breed**





Hint: ActionBar.getSelectedNavigationIndex() returns selected Tab index

Backward compatibility

- Old: ActionBarSherlock, which is built upon Support Library
- New: v7 appcompat library
- Note how to add library with resources:

  http://developer.android.com/tools/support-library/setup.html
- How to use:

  https://www.youtube.com/watch?v=6TGgYqfJnyc

Side Swipe

- ViewPager allows the user to flip left and right through pages of data – note we have a separate topic on Gestures (including more general swipe).
- In addition to ViewPager we will need PagerAdapter to generate the pages that the view displays.
- ViewPager is most often used in conjunction with Fragment, and views generated using either FragmentPagerAdapter (handful of typically more static fragments) or FragmentStatePagerAdapter (large number of pages).
- Note: ViewPager is under early design and development. The API will likely change in later updates of the compatibility library.

ViewPager Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <android.support.v4.view.ViewPager
        android:id="@+id/pager"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Activity

```
public class MainAct extends FragmentActivity {
    private MyAdapter mAdapter;
    private ViewPager mPager;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mAdapter = new MyAdapter(getSupportFragmentManager());
        mPager = (ViewPager) findViewById(R.id.pager);
        mPager.setAdapter(mAdapter);
    }

    private class MyAdapter extends FragmentPagerAdapter {

            ...
    }
}
```

Adapter

```
private class MyAdapter extends FragmentPagerAdapter {

    public MyAdapter(FragmentManager fm) {
        super(fm);
    }

    public int getCount() {
        return 3;
    }

    public Fragment getItem(int position) {
        switch (position) {
        case 0:
            return new DogFragment(R.drawable.borderterrier);
        case 1:
            return new DogFragment(R.drawable.rottweiler);
        case 2:
            return new DogFragment(R.drawable.dachshund);

        default:
            return null;
        }
    }
}
```

Fragment

```
public class DogFragment extends Fragment {

    private LinearLayout myView;
    private int resId;

    public DogFragment(int imageResourceId) {
        resId = imageResourceId;
    }

    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
{

        myView =(LinearLayout) inflater.inflate(R.layout.img, container, false);
        ImageView im = (ImageView) myView.findViewById(R.id.Img1);
        im.setImageResource(resId);
        return myView;
    }

}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schema..."
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <ImageView
        android:id="@+id/Img1"
        android:layout_width="450dp"
        android:layout_height="450dp"
        android:scaleType="centerCrop" />
</LinearLayout>
```

Navigation Drawer

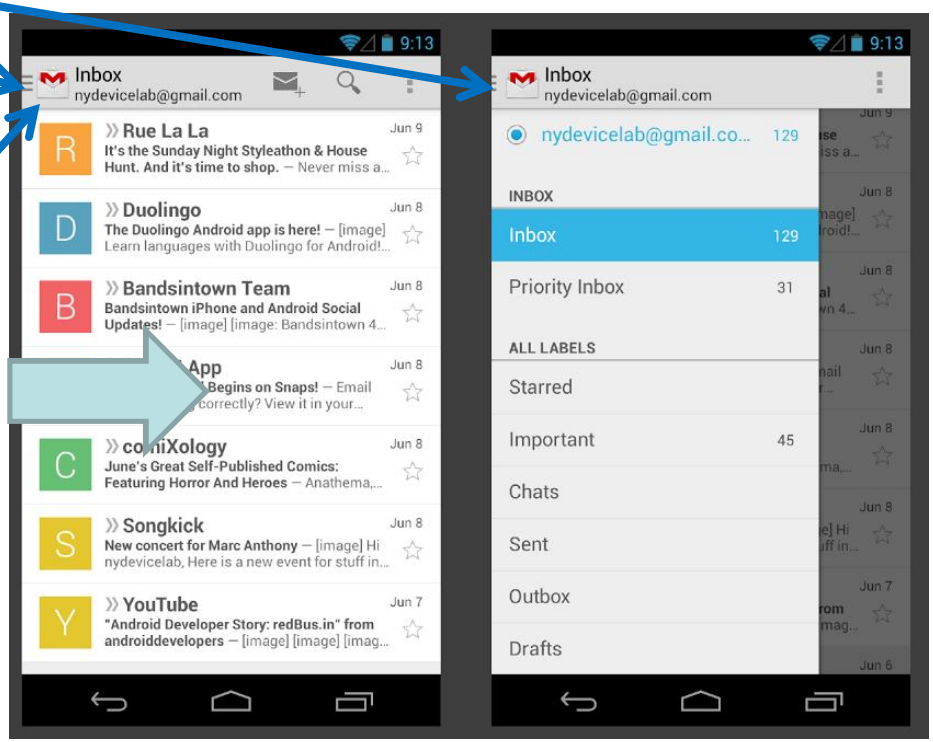
- Navigation Drawer is for chrome so it overlays app content – fast access to most popular content
- SlidingPaneLayout is for content navigation, thus it doesn't overlay content - master/detail pattern
- Switching between main use cases:
  - More than 3 top levels
  - Can include lower-level screens
  - Accessible from anywhere

https://www.youtube.com/watch?v=F5COhlbpIbY

http://www.androiduipatterns.com/2013/05/the-new-navigation-drawer-pattern.html

Navigation Drawer

- Own icon/glyph to indicate availability
- Start by swiping from the left edge or touching home icon



http://developer.android.com/training/implementing-navigation/nav-drawer.html

Create Navigation Drawer

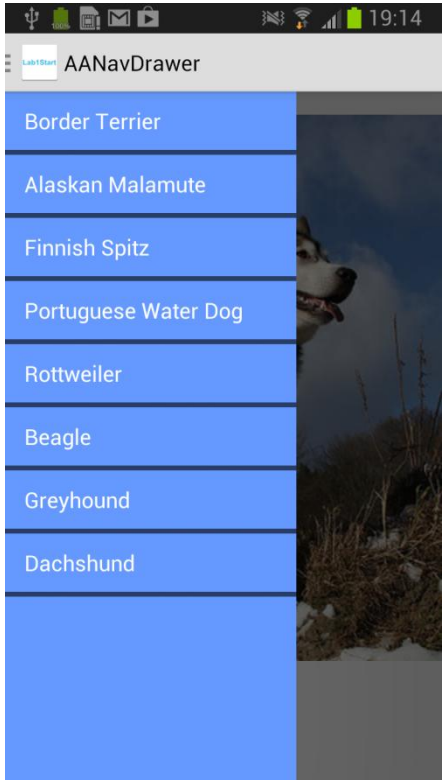
- Create DrawerLayout (part of support library): first view contains the main content and another view contains the contents of the navigation drawer (typically ListView).
- Initialize the navigation drawer's list of items - in case of ListView populate the list using an Adapter.
- Define and register OnItemClickListener to handle list item selections
- Define and register either DrawerLayout.DrawerListener or ActionBarDrawerToggle (if your activity includes the action bar) – ActionBarDrawerToggle contains DrawerListener and enables open and close Navigation Drawer by touching the app icon (app icon should also indicate the presence of the navigation drawer with a special icon)

Lab 5

- Read instructions from:

http://developer.android.com/training/implementing-navigation/nav-drawer.html

- Implement ”Dog Show”
  - Navigation Drawer
  - ActionBarDrawerToggle
  - Nav drawer icon
  - AALab1Start may help

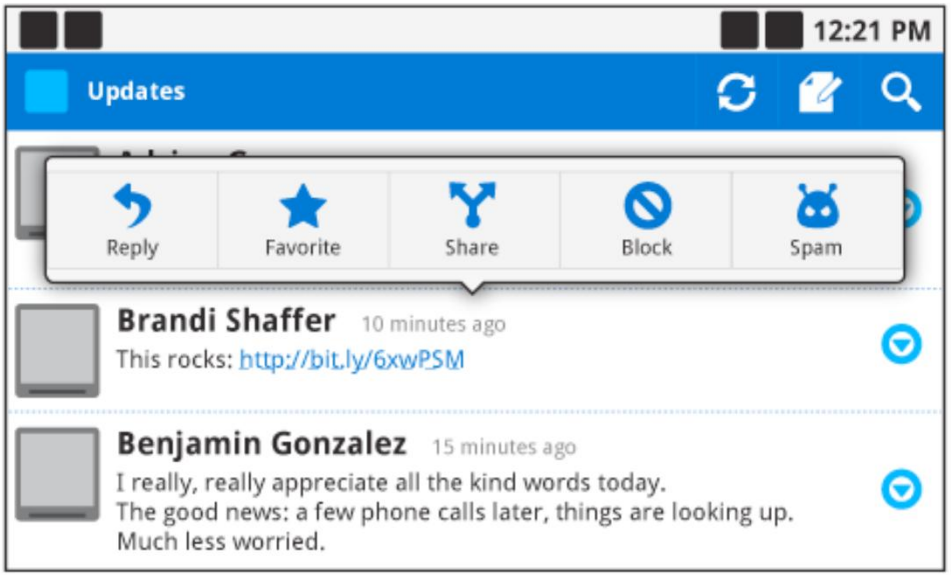



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/drawer_layout"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >
   <LinearLayout
     android:orientation="vertical"
     android:layout_width="match_parent"
     android:layout_height="match_parent">
     <TextView
            android:id="@+id/Text1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
    <ImageView
            android:id="@+id/Img1"
            android:layout_width="450dp"
            android:layout_height="450dp"
            android:scaleType="centerCrop" />
   </LinearLayout>
   <ListView
      android:id="@+id/left_drawer"
      android:layout_width="240dp"
      android:layout_height="match_parent"
      android:layout_gravity="left"
      android:divider="#293D66"
      android:dividerHeight="4dp"
      android:background="#6699FF"/>
</android.support.v4.widget.DrawerLayout>
```

Recap: Contextual Action Bar

- This has been covered already in Android Basic - course

Quick Actions



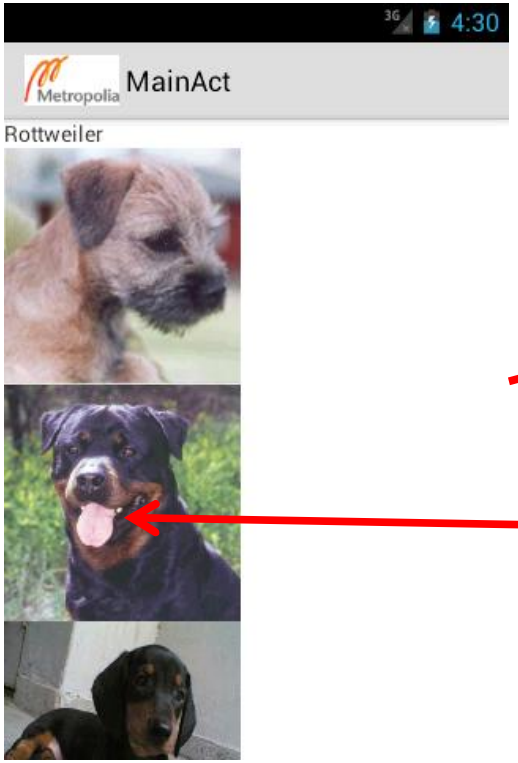
- QuickActions (as defined in Google 2010 IO / “Android UI Design Patterns”) is not included in standard Android SDK
- There are some examples how to build it from scratch:
  - http://www.londatiga.net/it/how-to-create-quickaction-dialog-in-android/
  - https://github.com/ruqqq/WorldHeritageSite/tree/master/src/sg/ruqqq/WHSFinder

Contextual Action Bar

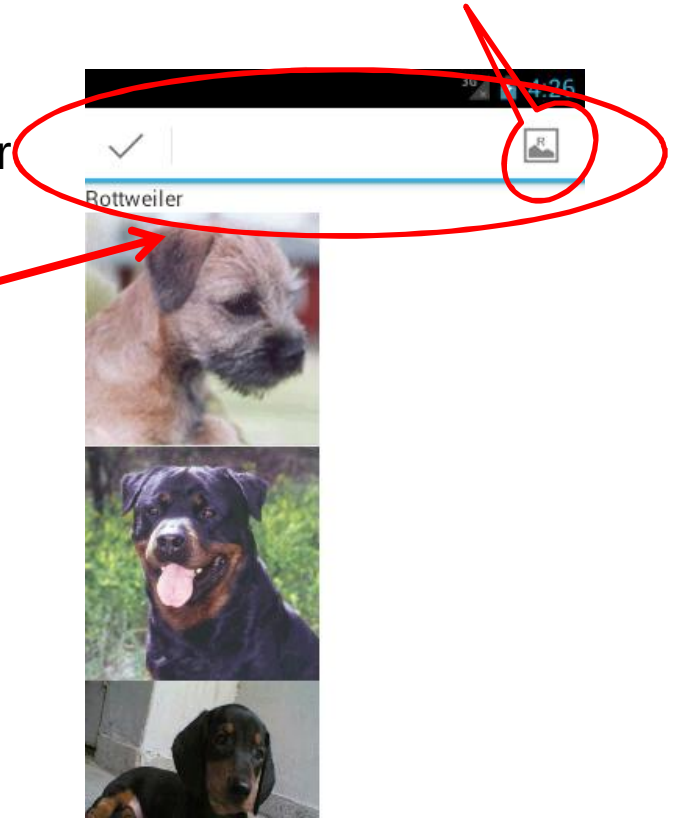
- When the user performs a long-click (press and hold) on a view:
  - Android 1.x – 2.x: contextual menu
  - Android 3.x - : contextual action mode
- Contextual action mode enables actions on selected content. This mode displays action / menu items that affect the selected content in a bar (Contextual Action Bar) at the top of the screen.
- Define menu items in xml-file
- Implement OnLongClickListener (start an action mode)
- Register listener to views
- Implement the ActionMode.Callback interface (contextual menu creation and menu item selection processing)

Dog Example

Action / menu item



Contextual Action Bar



Long click

Tuubi: AACAB1.zip

Main Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/Tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <ImageView
    android:id="@+id/Img1"
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:scaleType="fitXY" />
    <ImageView
    android:id="@+id/Img2"
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:scaleType="fitXY" />
    <ImageView
    android:id="@+id/Img3"
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:scaleType="fitXY" />
</LinearLayout>
```

Menuitems

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/dogb"
        android:icon="@drawable/ic_action_show"
        android:title="@string/menu_t"
        android:showAsAction="ifRoom|withText" />
</menu>
```

Activity

```
public class MainAct extends Activity {
    ActionMode mActionMode;
    int selB;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageView im = (ImageView)findViewById(R.id.Img1);
        im.setImageResource(R.drawable.borderterrier);
        im.setOnLongClickListener(myListener);
        im = (ImageView)findViewById(R.id.Img2);
        im.setImageResource(R.drawable.rottweiler);
        im.setOnLongClickListener(myListener);
        im = (ImageView)findViewById(R.id.Img3);
        im.setImageResource(R.drawable.dachshund);
        im.setOnLongClickListener(myListener);
    }
```

OnLongClickListener as inner class

```
private OnLongClickListener myListener = new OnLongClickListener(){
    public boolean onLongClick(View v){
            if (mActionMode != null) {
                return false;
                  }
            selB = v.getId();
            mActionMode = MainAct.this.startActionMode(mActionModeCallback);
            v.setSelected(true);
            return true;
      }
   };
```

ActionMode.Callback() Implementation as inner class

```
private ActionMode.Callback mActionModeCallback = new ActionMode.Callback() {

    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        MenuInflater inflater = mode.getMenuInflater();
        inflater.inflate(R.menu.mymenu, menu);
        return true;
    }

    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        return false;
    }

    public void onDestroyActionMode(ActionMode mode) {
        mActionMode = null;
    }
```

ActionMode.Callback() Implementation cont.

```
public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
    TextView tv = (TextView)findViewById(R.id.Tv1);
    String bText="";
    switch (selB){
        case R.id.Img1:
                bText = "Border Terrier";
                break;
        case R.id.Img2:
                bText = "Rottweiler";
                break;
        case R.id.Img3:
                bText = "Dachshund";
                break; }
    if (item.getItemId() == R.id.dogb ) {
        tv.setText(bText);
        mode.finish();  }
    return true;
}
```