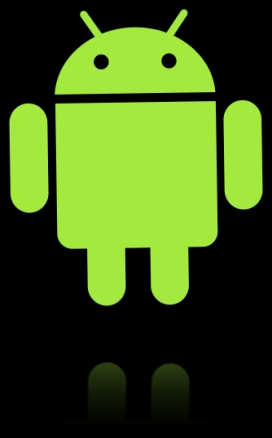
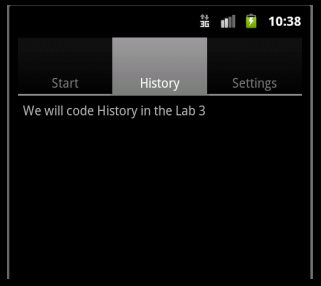


What you need to know for Lab 2

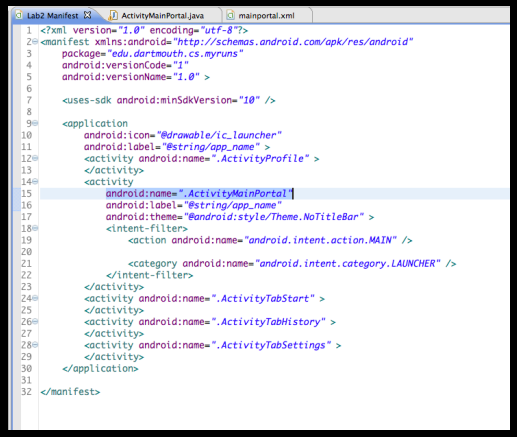
what we will cover

- setting up multiple activities
- intents
- lab 2

- multiple activities are used in lab 2 to manage different screens
- we need to set up different activities to handle different types of input on our UI



- Let's add a main activity and an activity for each tab in our UI to our Manifest file
- The main activity `android:name=".ActivityMainPortal"` will set up intents that fire if a tab is clicked by the user -- and the intents start the activity e.g., `android:name=".ActivityTabSettings"`



- Let's create a mainportal.xml and use a TabHost and a TabWidget to create a tabbed UI
- TabHost must be the root node for the layout
- It contains TabWidget for displaying the tabs and a FrameLayout for displaying the tab content.

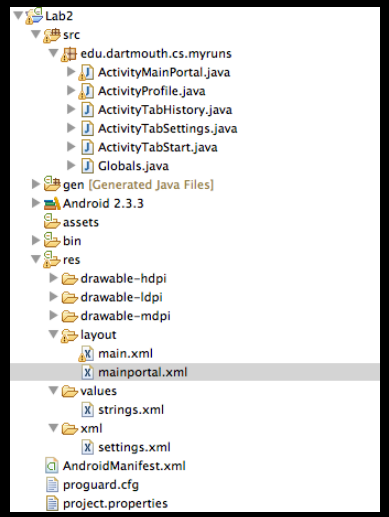
Lab2 Manifest | ActivityMainPortal.java | mainportal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="5dp">
        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp" />
    </LinearLayout>
</TabHost>
```

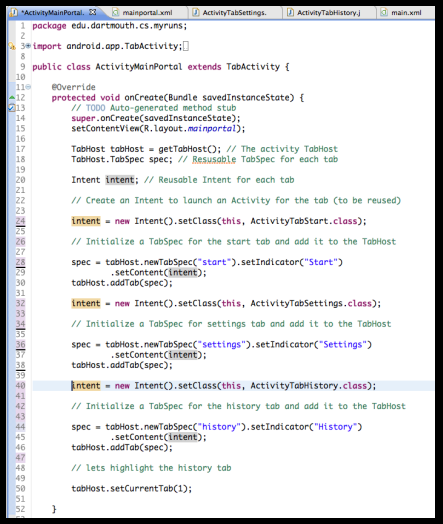
Quick look at our project for Lab 2. We will create the framework for the complete app on the UI side.

We will focus on allowing the user to enter richer information about themselves -- a picture from the camera, link in a webpage that can be fired up, use more widgets

Let's discuss the project layout of activities and various files and look at the code

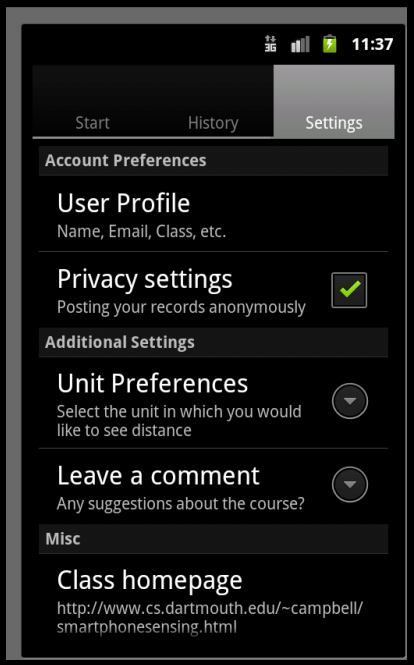


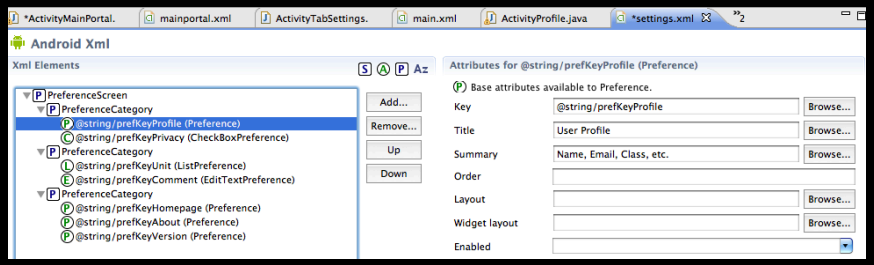
- Snippet of ActivityMainPortal
- It sets up each tab and creates an intent to fire the activity.
- Finally it renders the view.



```
package edu.dartmouth.cs.myruns;

import android.app.TabActivity;

public class ActivityMainPortal extends TabActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mainportal);

        TabHost tabHost = getTabHost(); // The activity TabHost
        TabHost.TabSpec spec; // Resusable TabSpec for each tab

        Intent intent; // Reusable Intent for each tab

        // Create an Intent to launch an Activity for the tab (to be reused)

        intent = new Intent().setClass(this, ActivityTabStart.class);

        // Initialize a TabSpec for the start tab and add it to the TabHost

        spec = tabHost.newTabSpec("start").setIndicator("Start")
                .setContent(intent);
        tabHost.addTab(spec);

        intent = new Intent().setClass(this, ActivityTabSettings.class);

        // Initialize a TabSpec for settings tab and add it to the TabHost

        spec = tabHost.newTabSpec("settings").setIndicator("Settings")
                .setContent(intent);
        tabHost.addTab(spec);

        intent = new Intent().setClass(this, ActivityTabHistory.class);

        // Initialize a TabSpec for the history tab and add it to the TabHost

        spec = tabHost.newTabSpec("history").setIndicator("History")
                .setContent(intent);
        tabHost.addTab(spec);

        // lets highlight the history tab

        tabHost.setCurrentTab(1);

    }
```

- The UI for settings looks like this.
- It includes categories that are grouped in settings.xml
-





- when looking at the settings.xml you can build your screen using the structure or xml views. In structure you can build your categories by adding PreferenceCategory to the PreferenceScreen.
- On the right you fill in the key, title and summary. The key is used to refer to the particular category for example prefKeyProfile.
- You can use that key to for example fire an activity if the category is selected by the user

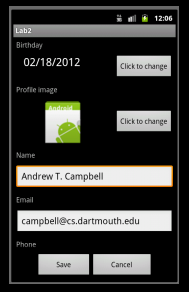
here is the settings.xml

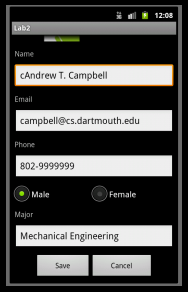
```xml
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >

    <PreferenceCategory android:title="Account Preferences" >
        <Preference
            android:key="@string/prefKeyProfile"
            android:summary="Name, Email, Class, etc."
            android:title="User Profile" />

        <CheckBoxPreference
            android:defaultValue="false"
            android:key="@string/prefKeyPrivacy"
            android:summary="Posting your records anonymously"
            android:title="Privacy settings" />
    </PreferenceCategory>
    <PreferenceCategory android:title="Additional Settings" >
        <ListPreference
            android:entries="@array/distanceMeasuredNameArray"
            android:entryValues="@array/distanceMeasuredValueArray"
            android:key="@string/prefKeyUnit"
            android:summary="Select the unit in which you would like to see distance"
            android:title="Unit Preferences" />

        <EditTextPreference
            android:key="@string/prefKeyComment"
            android:summary="Any suggestions about the course?"
            android:title="Leave a comment" />
    </PreferenceCategory>
    <PreferenceCategory android:title="Misc" >
        <Preference
            android:key="@string/prefKeyHomepage"
            android:summary="@string/projectURL"
            android:title="Class homepage" />
        <Preference
            android:key="@string/prefKeyAbout"
            android:title="About Runs" />
        <Preference
            android:key="@string/prefKeyVersion"
            android:summary="0.0.0"
            android:title="Version" />
    </PreferenceCategory>

</PreferenceScreen>
```

- ActivityTabSettings extends Preference Activity
- OnCreate sets up a listener that starts the ActivityProfile

```java
package edu.dartmouth.cs.myruns;

import android.app.AlertDialog;

public class ActivityTabSettings extends PreferenceActivity {

    private Context mContext;

    private final static int DIALOG_ABOUT_ID = 1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        mContext = this;

        Preference.OnPreferenceClickListener prefListerner;

        // instantiates preference object

        addPreferencesFromResource(R.xml.settings);

        // Get the usable preference
        Preference pref;

        // Setup listener for profile page to be renedered

        pref = findPreference(getString(R.string.prefKeyProfile));

        prefListerner = new Preference.OnPreferenceClickListener() {

            @Override
            public boolean onPreferenceClick(Preference preference) {

                startActivity(new Intent(mContext, ActivityProfile.class));
                return true;
            }

        };

        pref.setOnPreferenceClickListener(prefListerner);
```

If the user clicks on “User Profile” the on click listener starts ActivityProfile activity -- note the screen can scrowl

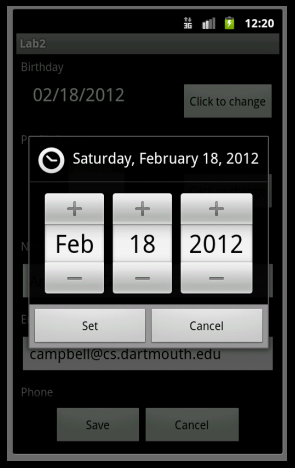




the main.xml (a better name would be profile.xml) uses ScrollView to scroll down or up through the screen

But the save and cancel buttons are always at the bottom of the screen. You can use the graphical and/or xml views to update the profile screen you created in Lab 1

Let’s quickly look at the main.xml



Picker dialog

```
<LinearLayout
    android:id="@+id/linearLayout3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:id="@+id/dateDisplay"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_weight="1"
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <Button
        android:id="@+id/buttonPickDate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="@string/buttonBirthday" />

</LinearLayout>
```

main.xml snippet

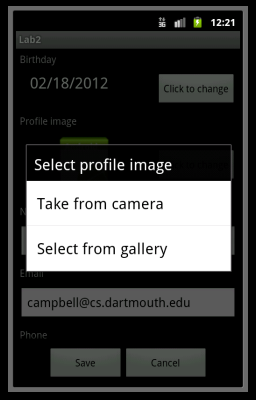
```
btn.setOnClickListener(myListener);

// handling the event of inputing the date -- clicking the date button

btn = ((Button) findViewById(R.id.buttonPickDate));

myListener = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        showDialog(DIALOG_BDATE_ID);
    }
};

btn.setOnClickListener(myListener);
```

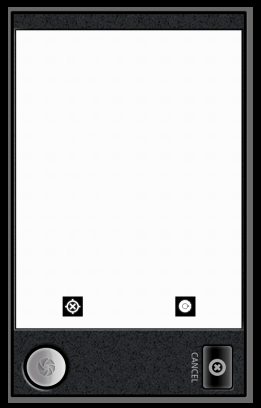
ActivityProfile.java snippet
create a dialog and listener

```
// when showDialog is executed it will execute this -- the ID defines what
// is done

@Override
protected Dialog onCreateDialog(int id) {

    // Override onDateSet method, save date into member variables and update
    // brithday textveiw
    switch (id) {
    case DIALOG_BDATE_ID:

        // if set is clicked on the picker date widget the onDateSet

        DatePickerDialog.OnDateSetListener myListerner = new DatePickerDialog.OnDateSetListener() {

            @Override
            public void onDateSet(DatePicker view, int year,
                    int monthOfYear, int dayOfMonth) {

                mYear = year;
                mMonth = monthOfYear;
                mDay = dayOfMonth;
                updateBirthdayDisplay();

            }
        };

        DatePickerDialog myDialog = new DatePickerDialog(this, myListerner,
                mYear, mMonth, mDay);
        return myDialog;
    }

    return super.onCreateDialog(id);
}
```

when showDialog is executed it will call onCreateDialog displaying the DatePickerDialog

You need to take care of a number of edge cases with birthday: need to store it, what about the first time through.



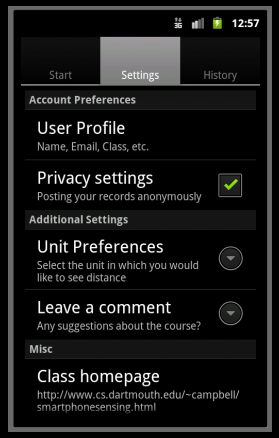


take a picture and store it in your profile. This is a tricky part of the lab

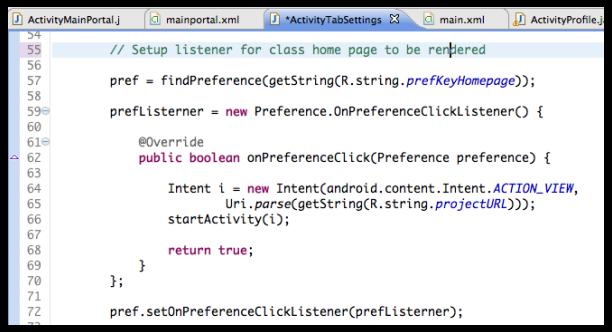
here a dialog is presented to the user

the ActivityProfile uses an intent to get a system activity to invoke the camera application to take a photo

the code needs to get return data from the camera activity (which is the photo). It needs to crop the photo to the right size, and save and render it in the profile.



ActivityTabSettings uses an intent to open a browser to display the class homepage



ActivityTabSettings uses an intent to open a browser to display the class homepage

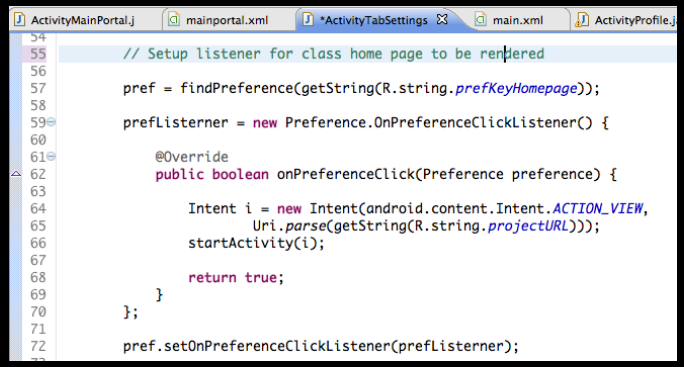
what have we discussed so far

- multiple activities
- intents and on click listeners
- preferenceActivity and category
- tabs, scrollviews
- dialogs
- firing system activities such as the camera and webpage

Let's discuss intents some more

intent

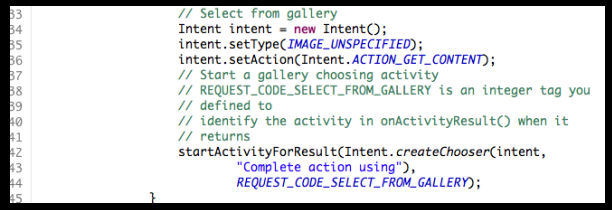
Intent is the glue that connects activities
- it allows one activity to link to another
- it allows you to pass between activities
- it allows you to call other activities e.g., our activities or built in apps/activities (camera, browser, etc.)



create an intent and then use startActivity to call the activity -- in this case the browser passing the URL as data **into** the browser activity

returning data from an activity

- startActivity() does not return data to the calling activity
- we have to use startActivityForResults() to do that.
- data may be passed into an activity (e.g., the web browser) or passed in and back; for example, in order for the camera activity to work it needs to pass data back to the calling activity (i.e., the photo)



```
// Select from gallery
Intent intent = new Intent();
intent.setType(IMAGE_UNSPECIFIED);
intent.setAction(Intent.ACTION_GET_CONTENT);
// Start a gallery choosing activity
// REQUEST_CODE_SELECT_FROM_GALLERY is an integer tag you
// defined to
// identify the activity in onActivityResult() when it
// returns
startActivityForResult(Intent.createChooser(intent,
        "Complete action using"),
        REQUEST_CODE_SELECT_FROM_GALLERY);
}
```

the galley activity needs to pass back the chosen picture to the activityProfile activity -- here is the setup with startActivityForResults()

```
// Handle date after activity returns -- camera and galley

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode != RESULT_OK)
        return;

    switch (requestCode) {
    case REQUEST_CODE_TAKE_FROM_CAMERA:
        // Send image taken from camera for cropping
        cropImage();
        break;

    case REQUEST_CODE_SELECT_FROM_GALLERY:
        // Send selected image from gallery for cropping
        mImageCaptureUri = data.getData();
        cropImage();
        break;

    case REQUEST_CODE_CROP_PHOTO:
        // Update image view after image crop
        Bundle extras = data.getExtras();
        // Set the profile image in UI
        if (extras != null) {
            mImageView
                    .setImageBitmap((Bitmap) extras.getParcelable("data"));
        }
        // Delete temporary image taken by camera after crop.
        File f = new File(mImageCaptureUri.getPath());
        if (f.exists())
            f.delete();
        break;
    }
}
```

You have to implement a onActivityResult method to get data back from the called activity; for example, the galley passes back a data object (the image) which we crop to size.

- Read chapter two in the book on Intents

- Read Intents and Intent filters on the dev site

http://developer.android.com/guide/topics/intents/intents-filters.html

lets look at the code